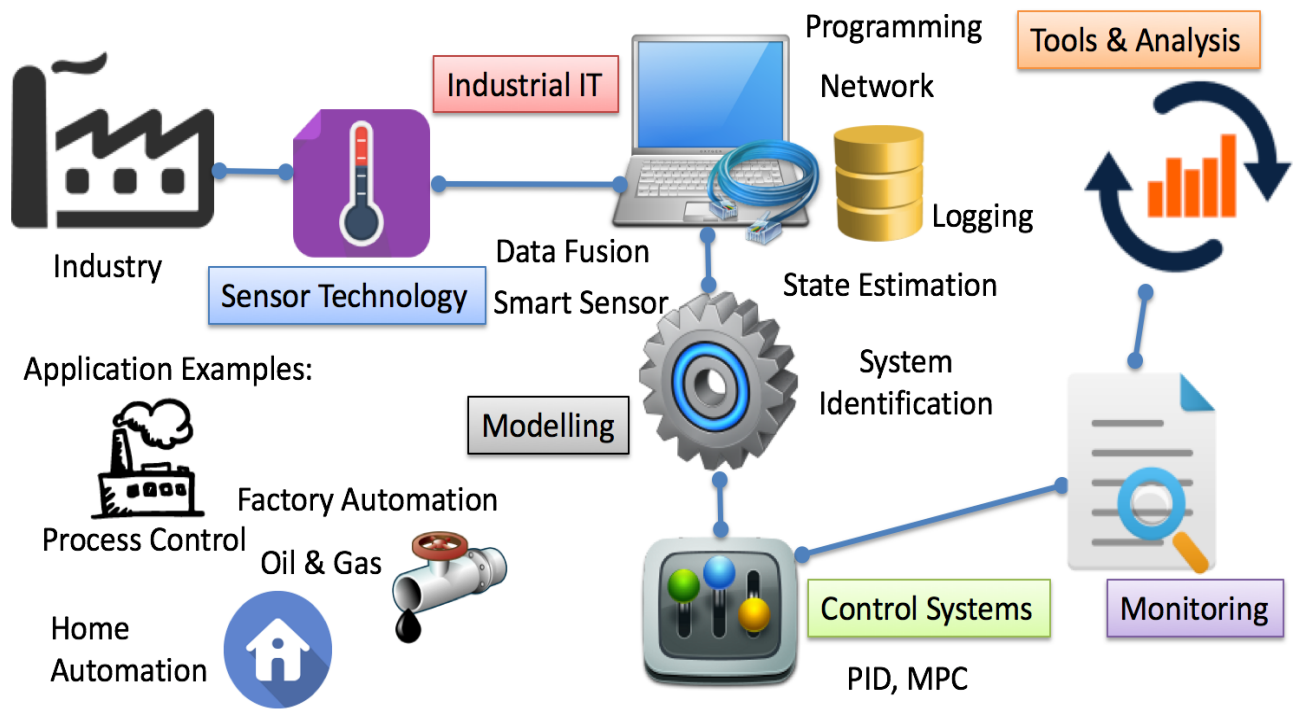


Industrial IT and Automation

A Practical Approach!

Hans-Petter Halvorsen



Industrial IT and Automation

A Practical Approach!

Hans-Petter Halvorsen

Copyright © 2017

ISBN: 978-82-691106-1-6

Publisher Identifier: 978-82-691106

<https://halvorsen.blog>

Preface

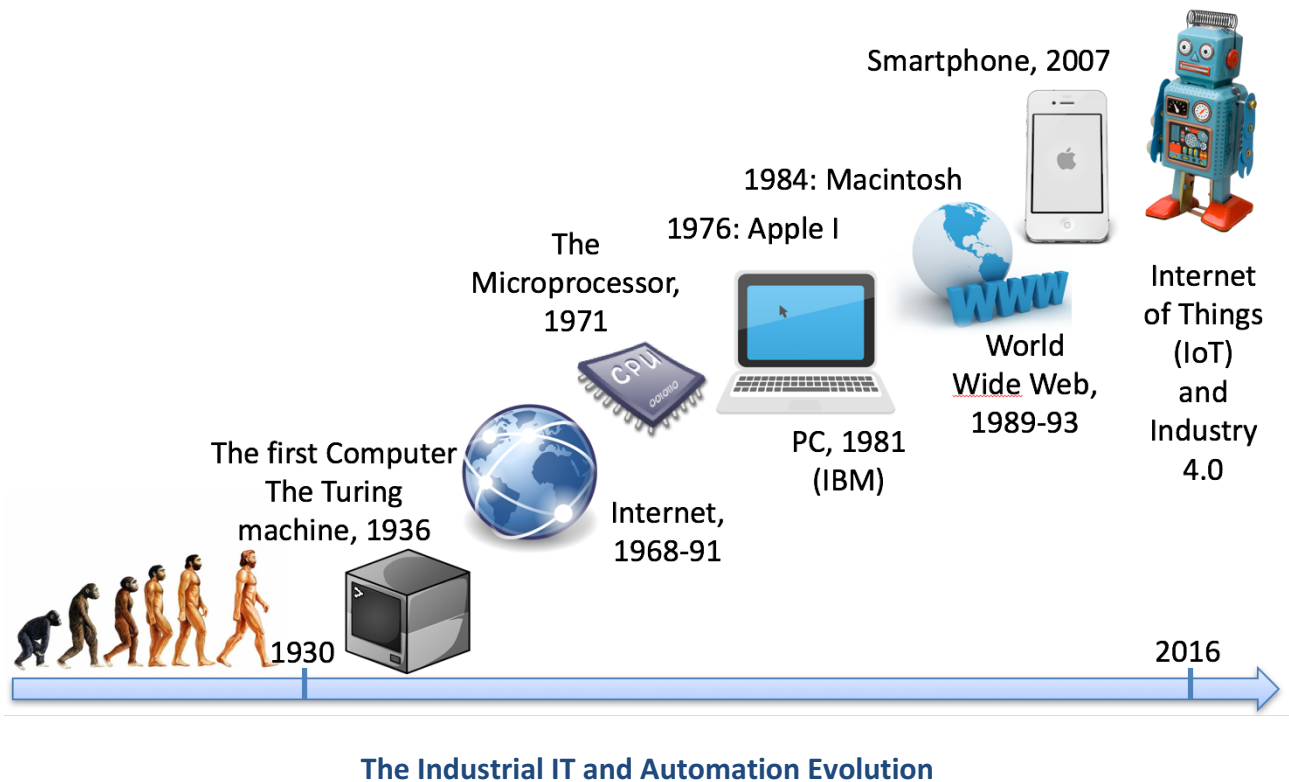
This document discusses different topics within Industrial IT and Automation with focus on practical examples. The different topics will introduce relevant software and different examples how to implement it in different programming languages like LabVIEW, Visual Studio/C# and MATLAB.

Industrial IT is the integration of Automation and Information Systems across the business. You could say Industrial IT is use of IT in industrial applications, everything from Process Control Systems, Sensor Technology, Data Acquiring, Data Logging and Monitoring and Software and Systems Engineering.

You need to have knowledge of Data Acquisition, Database Systems, Data Communication and Networks, Automation and Control, etc.

Terms such as Internet of Things (IoT), Smart Technology, Cloud Computing and Industry 4.0 are very popular these days. These topics and many others will be discussed in this document.

The figure below shows some important steps in the Industrial IT and Automation evolution:



This document discusses the following topics regarding Industrial IT and Automation (see figure below):



Word Cloud of some important Topics discussed in this document

This document discusses different topics within Industrial It and Automation with focus on practical examples.

The different topics will introduce relevant software and different examples how to implement it in different programming languages like LabVIEW, Visual Studio/C# and MATLAB.

Short overview of the contents:

- **Part 1:** System Engineering: Project Management, System Engineering, Software Development and Documentation. You need these tools in order to create professional Industrial IT and Automation Solutions.
- **Part 2:** Industrial IT: Data Communication, Database Systems, Web Services, Modbus, Virtualization, Wireless Systems.
- **Part 3:** Automation: DAQ, HMI, Control Systems, Sensors and Actuators, OPC, SCADA Systems, HIL Simulation.
- **Part 4:** Internet of Things: What is Internet of Things (IoT), Home Automation, Arduino, Raspberry Pi.
- **Part 5:** Applications and Examples: Here you find different applications and examples within different areas.

In addition to this document, you find lots of online resources, in-depth tutorials, PowerPoints videos, example codes, etc. on this web site:

<https://www.halvorsen.blog>

Information about the Author:

The author currently works at University College of Southeast Norway. The author has been working with Industrial IT and Automation Engineering for the last 20 years.



Hans-Petter Halvorsen

For more information, visit my web site:



<https://www.halvorsen.blog>

Table of Contents

Preface	iii
Table of Contents.....	vii
Part 1 System Engineering	15
1 Project Management	16
1.1 Project Planning	17
1.2 Kick-off/Brainstorming.....	17
1.3 Software Development Plan (SDP).....	18
1.3.1 Gantt Chart	18
1.4 Meetings	20
1.4.1 Notice of Meeting and Meeting Agenda.....	21
1.4.2 Minutes of Meeting	21
2 System Engineering.....	22
3 Requirements Engineering.....	25
3.1 UML.....	26
3.1.1 UML Software	27
3.2 Use Case	28
3.3 Sequence Diagram	29
3.4 Class Diagram	30
3.5 Database Design.....	30
4 Visual Studio Team Services.....	33
5 Software Architecture.....	37
5.1 Client-Server Architecture.....	37

5.2	3-Tier Architecture	38
5.2.1	3-Tier Architecture with Visual Studio	42
5.3	API	43
6	Implementation	46
6.1	LabVIEW	46
6.2	Visual Studio and C#.....	47
6.3	MATLAB.....	48
7	Testing.....	50
7.1	Levels of Testing.....	52
7.1.1	Unit Testing.....	53
7.1.2	Regression Testing.....	56
7.1.3	Integration Testing.....	57
7.1.4	System Testing/Validation Testing.....	57
7.1.5	Acceptance Testing.....	57
7.2	Test Categories.....	58
7.2.1	Black-box Testing.....	58
7.2.2	White-box Testing.....	58
8	Documentation	60
8.1	Process Documentation	63
8.2	Product Documentation.....	64
8.2.1	System Documentation	64
8.2.2	User Documentation.....	66
Part 2	: Industrial IT	69
9	Data Communication	70
9.1	Network	71
10	Database Systems	72

10.1	Structured Query Language (SQL)	73
10.2	SQL Server	75
10.3	ODBC	76
10.4	Database Communication in LabVIEW	77
10.4.1	LabVIEW SQL Toolkit.....	77
10.4.2	LabVIEW Example	78
10.5	Database Communication in C#	79
10.5.1	C# Example	79
11	Web Services.....	81
11.1	Web Services with LabVIEW	83
11.2	Data Dashboard for LabVIEW	89
12	Modbus.....	92
12.1	What is Modbus?.....	92
12.2	Modbus Register Types	93
12.2.1	Access Levels.....	94
12.3	Modbus Protocols	94
12.3.1	Modbus ASCII.....	95
12.3.2	Modbus RTU	95
12.3.3	Modbus TCP/IP	95
12.4	Modbus in LabVIEW	95
12.4.1	LabVIEW Modbus API	96
12.4.2	LabVIEW Modbus Simulator	99
13	Virtualization.....	100
13.1	Introduction.....	100
13.2	VMware	101
13.2.1	VMware Workstation Player.....	101

13.2.2	VMware Workstation	102
13.2.3	VMware vSphere	102
13.3	Microsoft Hyper-V	102
13.3.1	Windows Server with Hyper-V	103
13.3.2	Hyper-V Server	103
13.3.3	Windows Client Hyper-V	103
13.4	VirtualBox	105
13.5	Mac and OS X	105
13.5.1	BootCamp	105
13.5.2	VMware Fusion	106
13.5.3	Parallels Desktop	106
13.5.4	VirtualBox	106
14	Cloud Computing	108
14.1	Windows Azure	109
14.2	Amazon Web Services	109
14.3	Google Cloud Platform	110
15	Wireless Systems	111
15.1	ZigBee	112
15.2	WirelessHART	112
16	Vision Systems	113
16.1	Vision Systems in LabVIEW	113
16.2	Vision Systems in Visual Studio/C#	114
Part 3 : Automation		115
17	DAQ Systems	116
17.1	Sampling	117
17.1.1	AD Converters	118

17.2	DAQ Hardware.....	119
17.2.1	NI USB TC-01 Thermocouple Device	119
17.2.2	NI USB-6008 DAQ Device	120
17.3	NI DAQmx driver.....	121
17.3.1	NI MAX.....	123
17.4	Measurement Studio.....	123
17.5	DAQ with LabVIEW	124
17.5.1	Write to DAQ	126
17.6	Datalogging.....	127
17.6.1	Measurement Filter/Low-pass Filter	129
17.6.2	LabVIEW Example	130
17.6.3	C# Example	133
17.7	Industrial DAQ Systems	134
17.7.1	cDAQ.....	134
17.7.2	cRIO.....	134
18	User Experience	136
19	Control Systems	137
19.1	PC-based Control System	137
19.2	PID Control	138
19.2.1	PI Controller as a State-space model	140
19.3	Industrial Control Systems.....	141
20	Sensors and Actuators	143
20.1	Sensors	143
20.1.1	Pt-100	145
21	OPC	150
21.1	What is OPC?	150

21.1.1	OPC Specifications	151
21.2	MatrikonOPC Simulation Server	152
21.2.1	MatrikonOPC Explorer (OPC Client)	152
21.3	OPC DA in LabVIEW	153
21.3.1	Write to OPC Server using LabVIEW	154
21.3.2	Read from OPC Server using LabVIEW	155
21.4	OPC DA in Visual Studio/C#	155
21.4.1	Read OPC Data	156
21.4.2	Write OPC Data	158
21.4.3	Using a Timer	160
21.5	OPC DA in MATLAB	161
21.6	OPC UA	161
21.6.1	OPC UA in LabVIEW	163
21.6.2	OPC UA in MATLAB	165
22	SCADA Systems	166
22.1	Introduction	166
23	HIL Simulation	168
23.1	What is HIL Simulation?	168
23.2	Why use HIL simulation?	169
23.3	Challenges	171
23.4	Applications	171
23.4.1	Embedded Control Systems	171
23.5	Procedure	172
23.6	Practical Example	172
23.6.1	Introduction	172
23.6.2	Simulated Process	173

23.6.3	Hardware	174
23.6.4	The Procedure.....	174
23.6.5	HIL Simulation in LabVIEW.....	176
Part 4 : Internet of Things		178
24	Internet of Things (IoT)	179
24.1	Data Logging.....	181
24.1.1	Web-based Logging Services.....	181
25	Home Automation	183
25.1	Home Automation Platform	185
26	Arduino	188
26.1	Arduino UNO	188
26.2	Sensors and Actuators	189
26.3	Software	190
26.4	Code Examples	192
26.4.1	TMP36 Temperature Sensor Example	193
26.4.2	NTC Thermistor Example	196
26.5	Arduino Shields.....	199
26.6	XBee.....	200
26.7	XBee Hardware.....	201
26.8	Fritzing.....	203
27	Raspberry Pi	204
27.1	Accessories	207
27.2	Communication Protocols	207
27.3	Windows 10 IoT Core	207
28	Industry 4.0.....	208
Part 5 : Applications and Examples		214

29	Weather Station.....	215
29.1	Database.....	217
29.2	OPC Server.....	218
29.3	Web Service.....	218
29.4	iPad App	219
29.5	Windows 10 Universal App	220
30	DeltaV Training and Research Center	221
30.1	Training Center	222
30.2	Research Center	223
31	Data Management and Monitoring Platform	225
	References	228

Part 1 System Engineering

In this part, we start with an introduction to Project Management and System Engineering. Since this document focus on practical implementations, it is crucial that proper project management is followed and that you use a system engineering approach when you develop your solutions.

Web: https://www.halvorsen.blog/documents/programming/software_engineering/

1 Project Management

Since this document focus on practical implementations, it is crucial that proper project management is followed and that you use a system engineering approach when you develop your solutions.

Project management is the key factor in any software development projects. Project management is the discipline of planning, organizing, motivating, and controlling resources to achieve specific goals (Halvorsen, Hans-Petter (2017). *Software Development - A Practical Approach!*).

In Figure 1-1 we see the well-known project triangle.

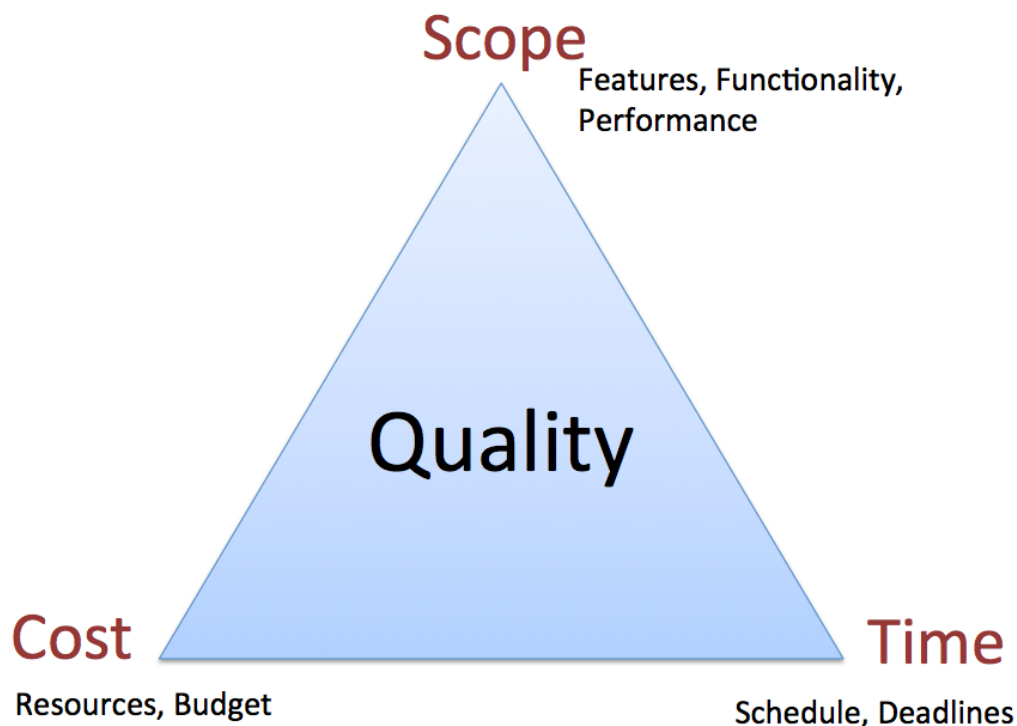


Figure 1-1: Project Triangle

Here are some Key factors to success:

- Kick-off and Brainstorming
- Planning and Estimation
- Project Tracking
- Communication and Collaboration
- Meetings
- Using proper Tools, such as e.g., Visual Studio Online

1.1 Project Planning

Software development involves lots of activities that need to be planned and synchronized. In order to do that we need good tools for these activities. The Gantt chart is probably the most used tool. In addition, we need to have different meetings in order to plan and coordinate the different activities.

1.2 Kick-off/Brainstorming

A Project should always start with a Kick-off meeting where a brainstorming session is important of that meeting.

During the brainstorming, you should:

- Involve all in the group
- Discuss what you are going to do in the project
- How are you going to solve the project?
- etc.



<http://geek-and-poke.com>

In addition to get good ideas for solving the project, you should learn from previous projects.

Examples: Who are going to solve the different parts, what kind of Frameworks are you going to use, what kind of development tools shall you use, etc.

1.3 Software Development Plan (SDP)

A good idea is to create a Software Development Plan. The Software Development Plan gives an overview of all the communication within the project or within the team, i.e., what kind of communication, how the communication should be done, etc.

Examples of Communication:

- Meetings: The Team will meet every Monday from ...
- Standards: Which Word processor, Templates, etc.
- E-mail... or other communication platforms, ...
- Collaboration: How will you communicate? Work together on Tuesdays, ...
- Other Tools: Microsoft Project, ...
- etc.

The Software Development Plan typically includes the following sections:

1. **Introduction:** This briefly describes the objectives of the project and set out the constraints (e.g., budget, time, etc.) that affects the management of the project
2. **Project Organization:** This section describes how the development team is organized, the people involved and their roles in the team.
3. **Risk Analysis**
4. **Hardware and Software Resource Requirements**
5. **Work Breakdown (WBS, Work Breakdown Structure):** Break down the project in into activities and identifies milestones
6. **Project Schedule:** Shows dependencies between activities, the estimated time required to reach each milestone, allocation of people to activities. (5) and (6) is typically done in a Gantt Chart (created in e.g. Microsoft Project)
7. **Monitoring and Reporting Mechanisms:** Definition of the Management Report that should be produced, when these should be produced, etc.

Other words for the Software Development Plan may be “Communication Plan” or “Project Plan”.

1.3.1 Gantt Chart

One of the most used tool for project planning is the Gantt chart. The Gantt chart gives an overview of tasks, subtasks, milestones, resources, etc. in a project.

In Figure 1-2 we se a Gant Chart example created with Microsoft Project.

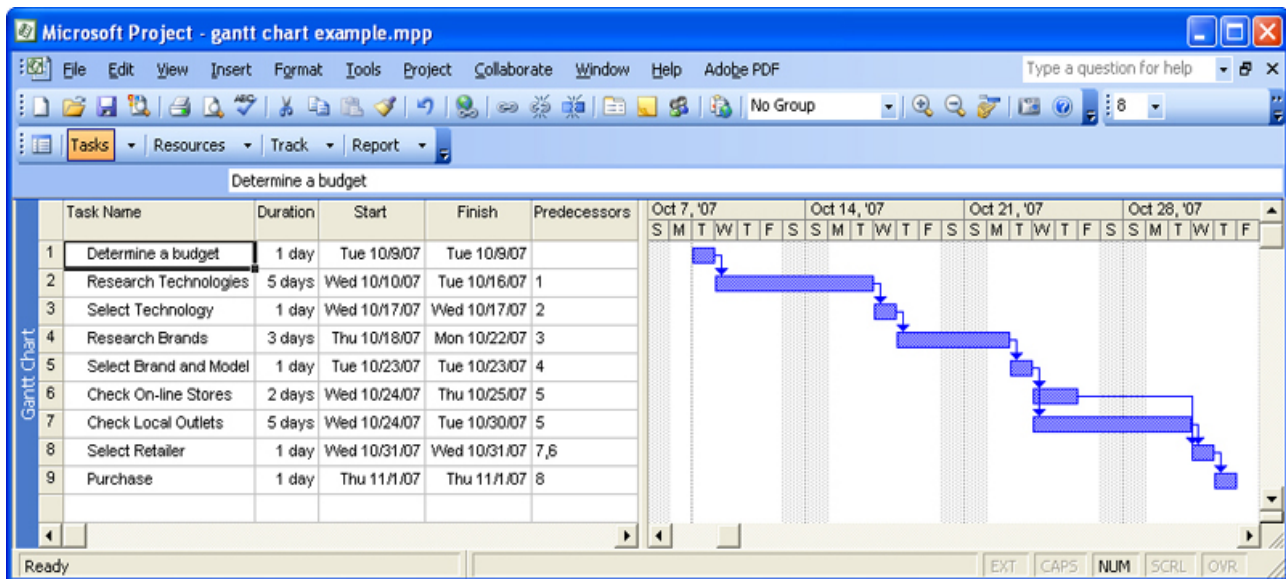
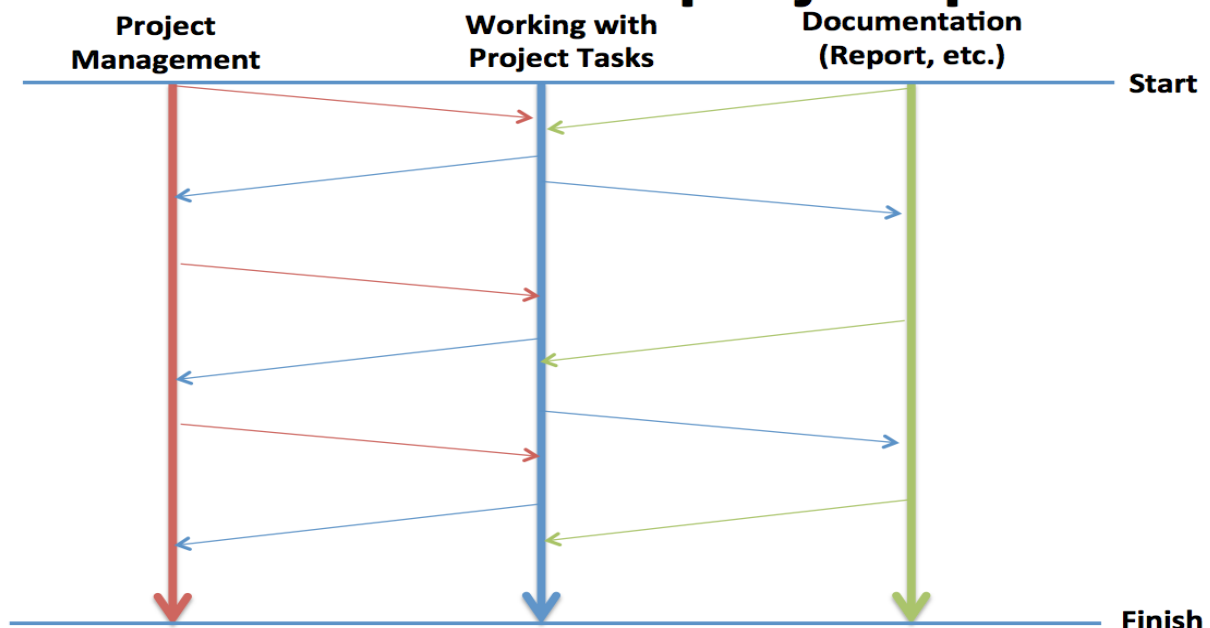


Figure 1-2: Gant Chart Example – Microsoft Project

It is important that the Project Management is an active part of your software project. The Gantt Chart should be used through the whole project, it is not something you create in the beginning of the project and put in a drawer.

In Figure 1-3 we see the recommended way of working with the different project activities.

How to work in the project period



Important: Work with these activities in parallel!!!

Figure 1-3: Project Work

1.4 Meetings

It is necessary to have meetings when planning and creating software, but these meetings should not be misused.

Below we list some typical meeting needed during the software development project:

- Kickoff and Planning Meetings
- Project Meetings
- Daily Scrum Meetings
- Review Meetings
- Meetings for Planning next Sprint/Iteration
- etc.

For meetings in general we have the following guidelines:

- The meeting agenda should be clear.
- All meetings should follow the basic structure that is described for that meeting.
- Meetings should start on time, even if some team members are late.
- Meetings should finish on time.
- Each team member should come to the meeting prepared.

1.4.1 Notice of Meeting and Meeting Agenda

A typical meeting agenda could be as follows:

- Project Plan, Gantt Chart (Project Manager)
- Work Items, Overview and Status (Test Manager)
- Demonstration of Applications/Coding (Individual)
- Short Status for each member (Individual)
 - What have you done so far?
 - What shall be your main focus the next weeks?
 - Any Technical Challenges/Problems/Issues? (It is very important to get an overview of the challenges in the project, or else the whole project will be at risk if you don't tell about them!)
 - Other matters
- The meeting should last no longer than 60 minutes.

When you are finished with the meeting, write a short Minutes of Meeting as soon as possible.

1.4.2 Minutes of Meeting

Write a “Minutes of Meeting” (send on e-mail to team members and supervisor the same day!).

The purpose of this is twofold:

- Important decisions or agreements are recorded, so they are not forgotten!
 - The second purpose is to record unsolved issues that require follow up action, so-called action items. Each action item is assigned to one (preferred) or more team members with a specific deadline for completion.
-

2 System Engineering

What is System Engineering? It is a complex process to develop modern and professional systems today (Halvorsen, Hans-Petter (2017). *Software Development - A Practical Approach!*).

System Engineering: The process of analyzing and designing an entire system, including the hardware and the software.

Software Engineering: The discipline for creating software applications. A systematic approach to the design, development, testing, and maintenance of software.

A lot of systems today have a mix of hardware and software that is tightly integrated, like modern smartphones, tablets, etc. To create such systems involves a lot of different disciplines.

Software, is any set of machine-readable instructions that directs a computer's processor to perform specific operations. The term is used to contrast with computer hardware, the physical objects (processor and related devices) that carry out the instructions. Computer hardware and software require each other and neither can be realistically used without the other, see Figure 2-1.

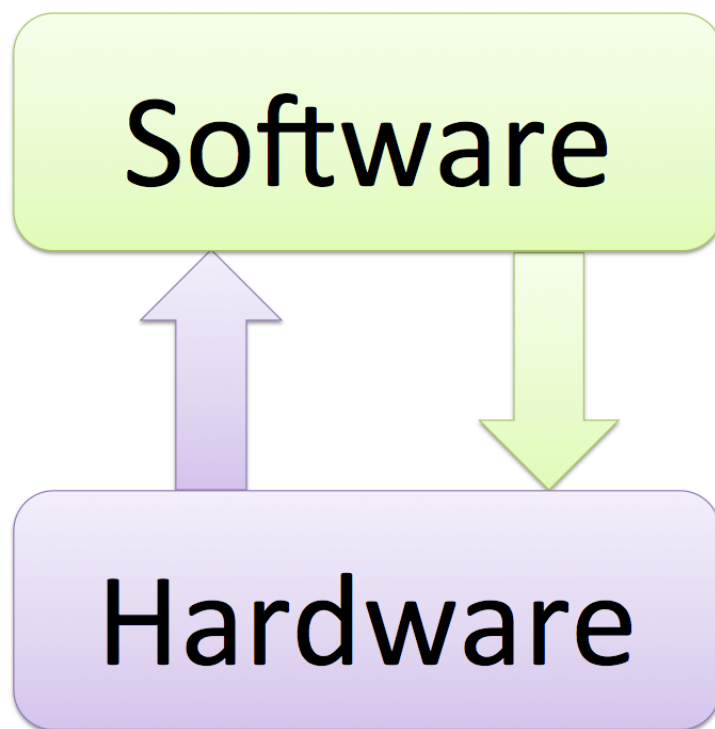


Figure 2-1: Hardware and Software working together

In Figure 2-2 we see a typical network and infrastructure that is part of the system.

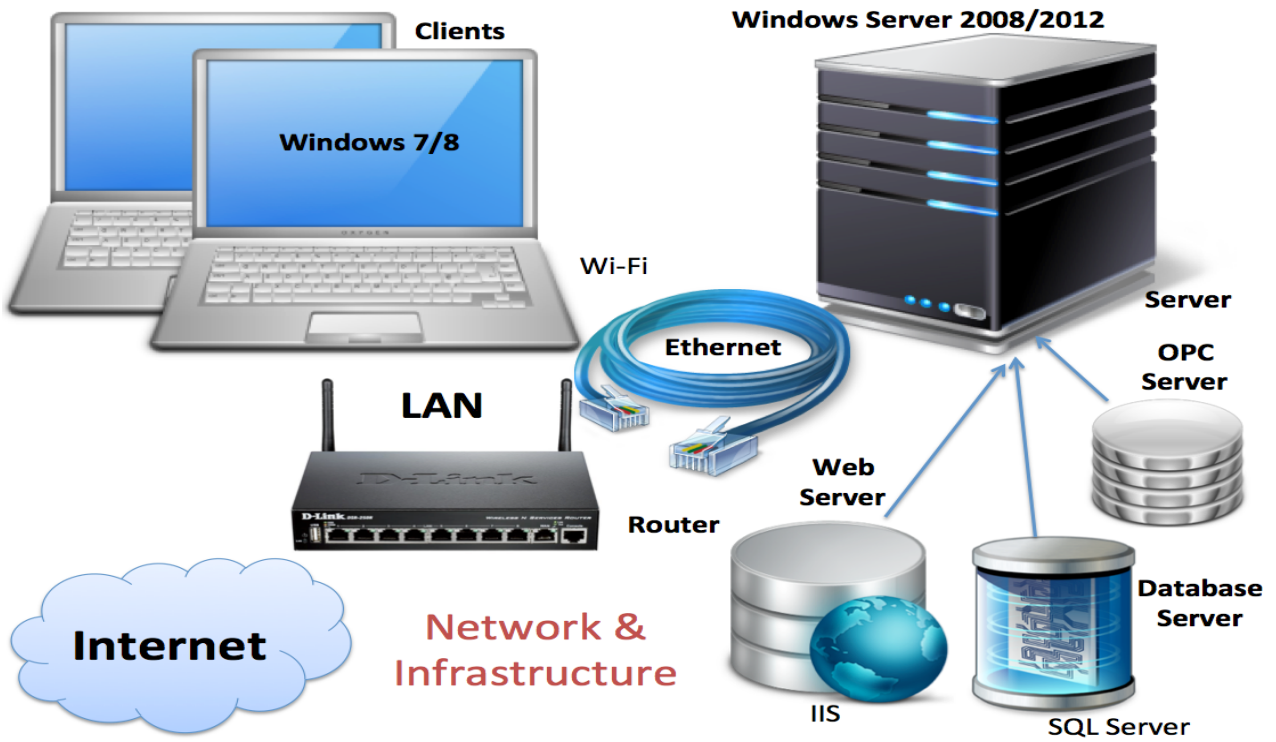


Figure 2-2: Typical Network and Infrastructure in Software Development

In Figure 2-3 we see the different phases involved in the Software Development Lifecycle (SDLC).

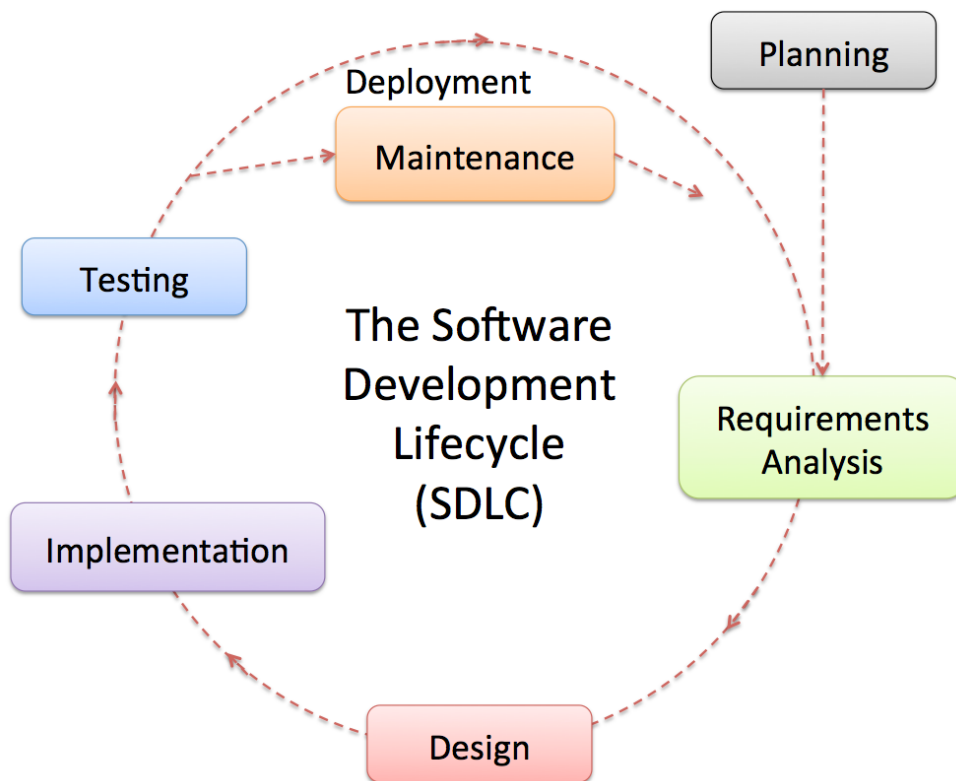


Figure 2-3: The Software Development Lifecycle

The main parts or phases in the software development process are:

- Planning
 - Requirements Analysis
 - Design
 - Implementation
 - Testing
 - Deployment and Maintenance
-

3 Requirements Engineering

Before you start to implement a software system, you need to understand what the system is intended to do. This intended functionality is the “Requirements”. The process of creating these requirements is called Requirement Analysis or Requirement Engineering. It is the process of understanding what you want and what you need in your software.

What is Requirements Engineering? Requirements is the bridge between the real world and the software system (Figure 3-1).

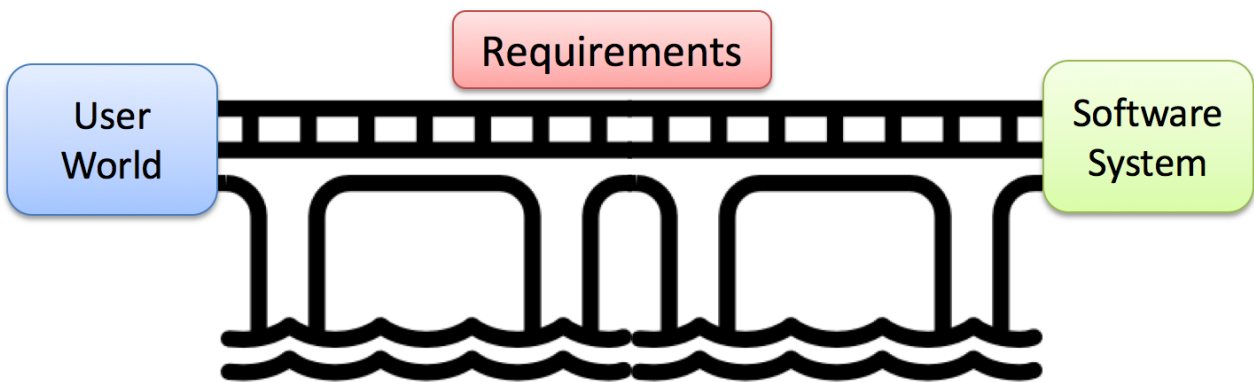


Figure 3-1: Requirements Engineering

You should create Design and Specifications (including UML) before you start Coding. UML diagrams is a general method/standard to do just that. But UML can also be used to document your code afterwards (so-called Reverse Engineering).

Figure 3-2 shows an example why Requirements Engineering is needed.

Requirements Engineering



What the Customer got



What the Customer really needed

Figure 3-2: Why Requirements Engineering is needed

3.1 UML

UML is a modeling language used in software engineering. It is very popular within OOA, OOD, OOP. UML was developed in the 1990s and adapted as an ISO standard in 2000. UML 2.2 has 14 different types of diagrams.

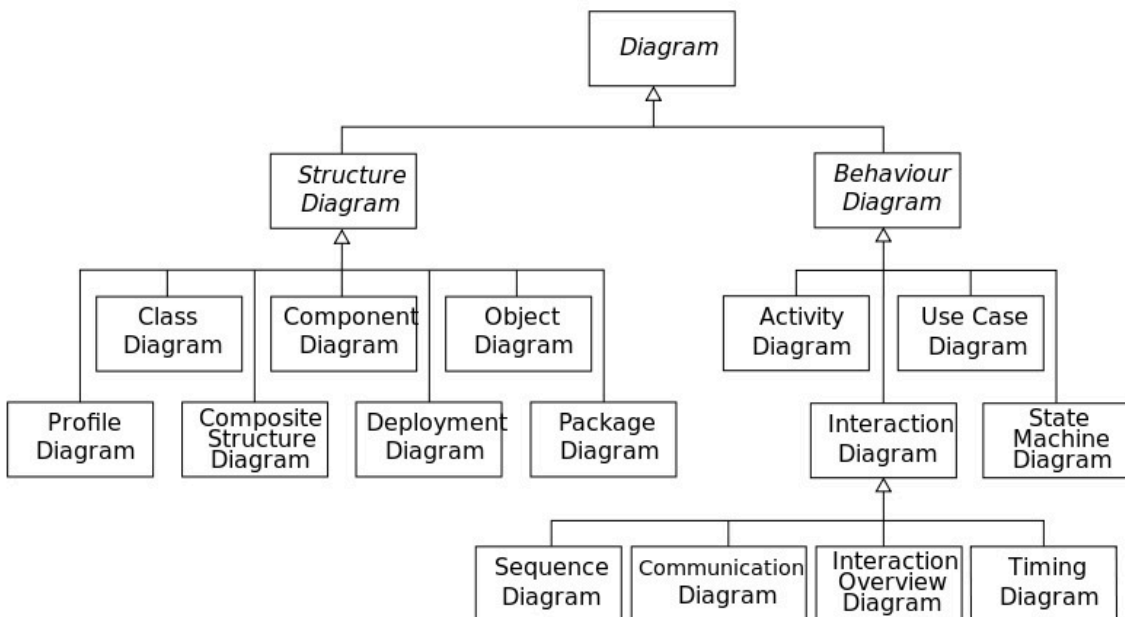


Figure 3-3: UML Diagrams

We have 2 main categories of diagrams:

- Structure Diagrams
- Behavior Diagrams
 - Interaction Diagrams (sub category of Behavior Diagrams)

The diagrams available in UML are:

- **Class Diagram**
- Component Diagram
- Deployment Diagram
- Object Diagram
- Package Diagram
- Activity Diagram
- **Sequence Diagram**
- Communication Diagram
- **Use Case Diagram**
- **State Machine Diagram**
- Composite Structure Diagram
- Interaction Overview Diagram
- Timing Diagram

Why use UML?

- Design:
 - Forward Design: doing UML before coding. Makes it easier to create the code in a structured manner
 - Backward Design: doing UML after coding as documentation
- Code

3.1.1 UML Software

There exists hundreds of different software for creating UML diagrams, here I mention just a few:

- Visio
- Enterprise Architect
- Visual Studio (Enterprise)

With the Visual Studio Enterprise edition, we can create some of the most used UML diagrams, see Figure 3-4.

These diagrams are available from the “Architecture” menu in Visual Studio.

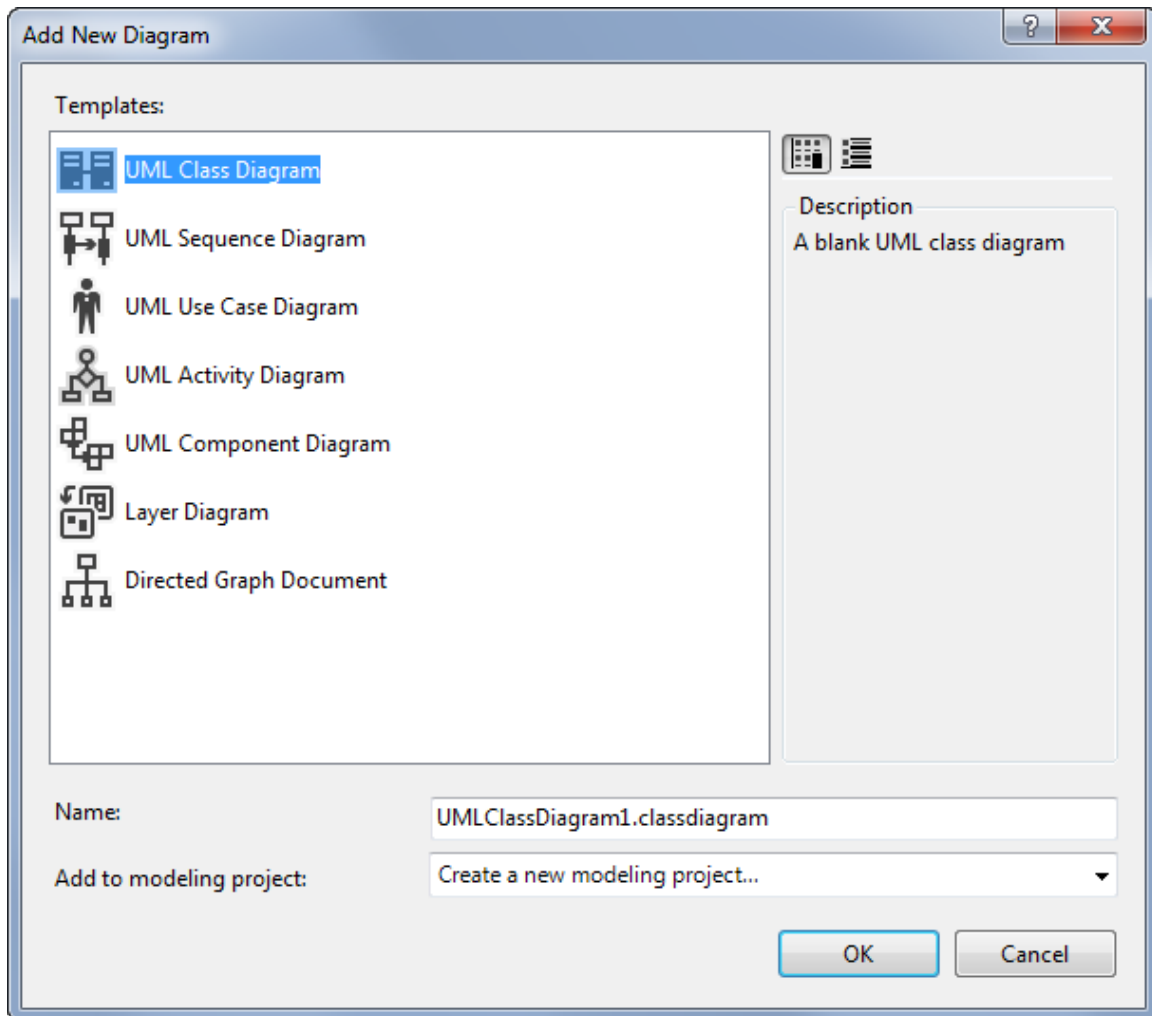


Figure 3-4: Create UML diagrams with Visual Studio Enterprise

3.2 Use Case

One of the most used UML diagrams is the Use Case Diagram.

In Figure 3-5 we see a Use Case example.

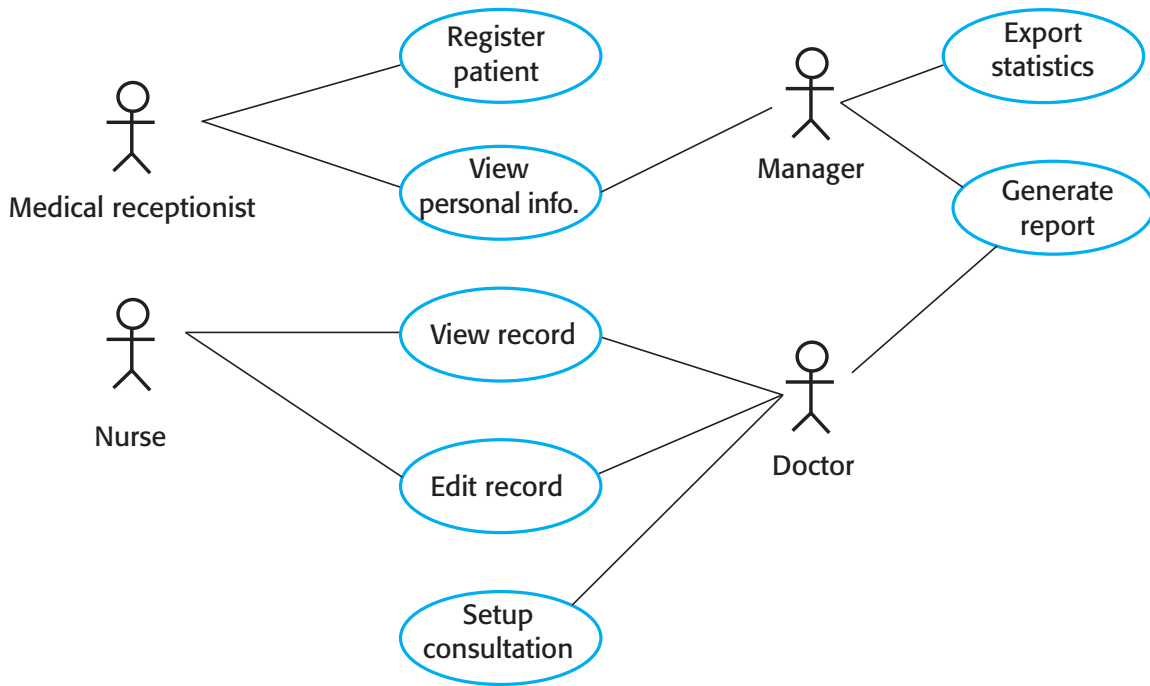


Figure 3-5: Use Case example

3.3 Sequence Diagram

In Figure 3-6 we see an example of a Sequence Diagram.

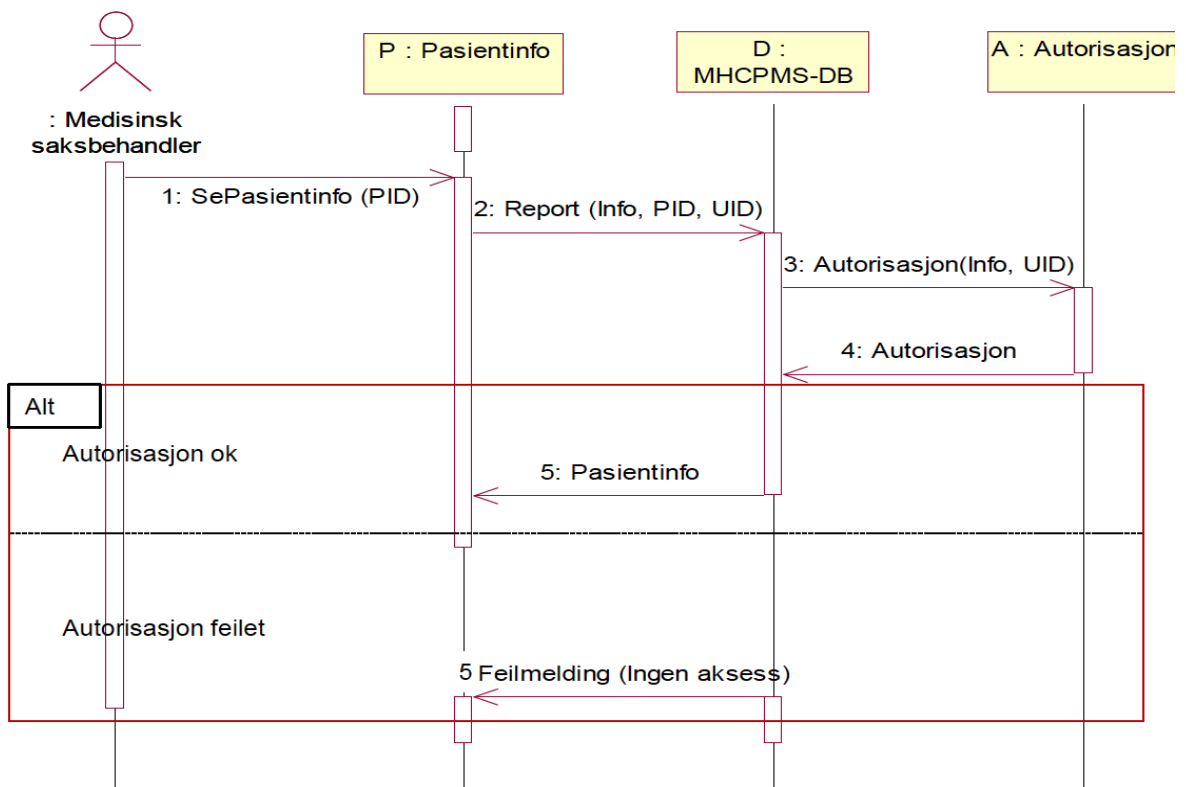


Figure 3-6 Sequence Diagram Example

3.4 Class Diagram

Figure 3-7 shows a Class Diagram Example.

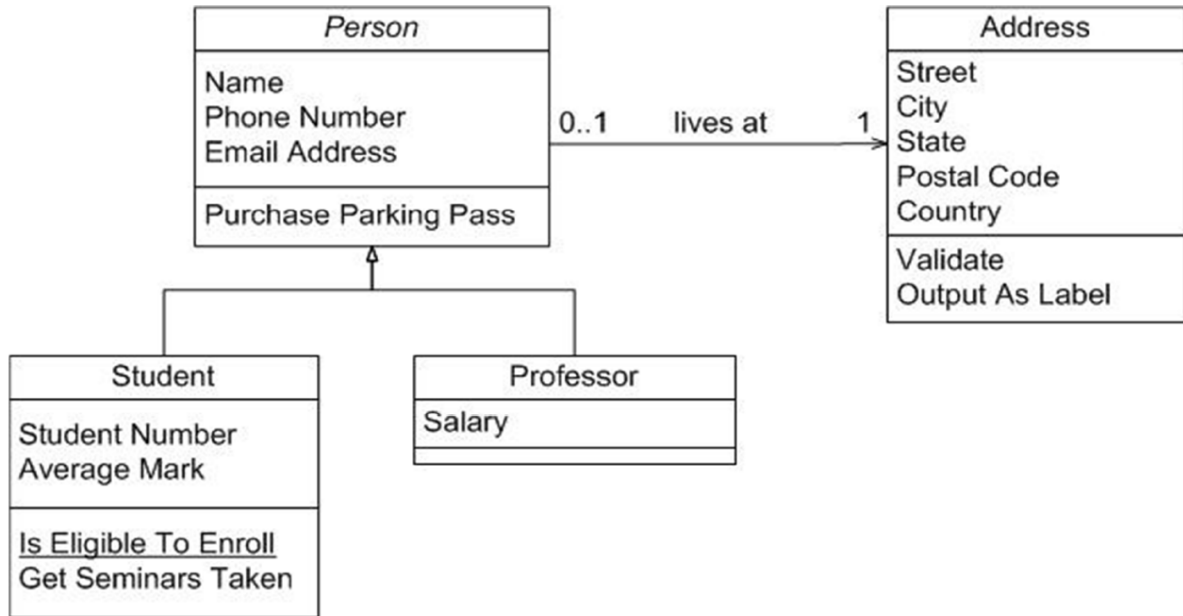


Figure 3-7: Class Diagram Example

3.5 Database Design

ER Diagram (Entity-Relationship Diagram) is used for design and modeling of databases. It specifies tables and relationship between them (Primary Keys and Foreign Keys), see Figure 3-8.

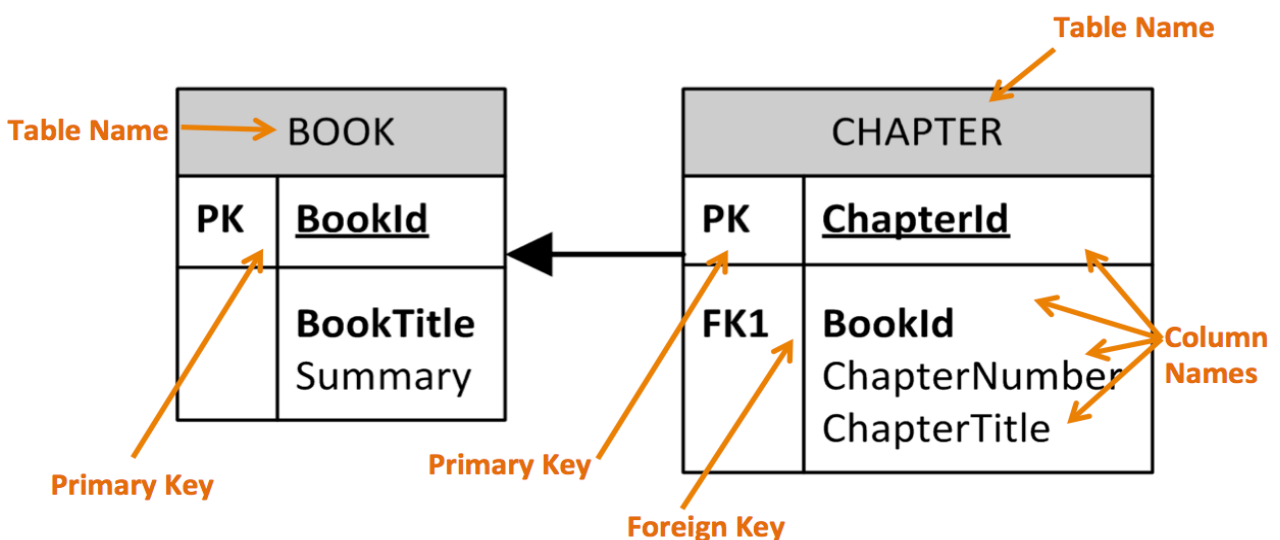


Figure 3-8: ER diagram with Primary Keys and Foreign Keys relationships

We can use a lot of different tools to create such ER diagram. In this document I will only focus on the following:

- MS Visio
- ERwin

In Figure 3-9 we see a typical ER diagram created in MS Visio.

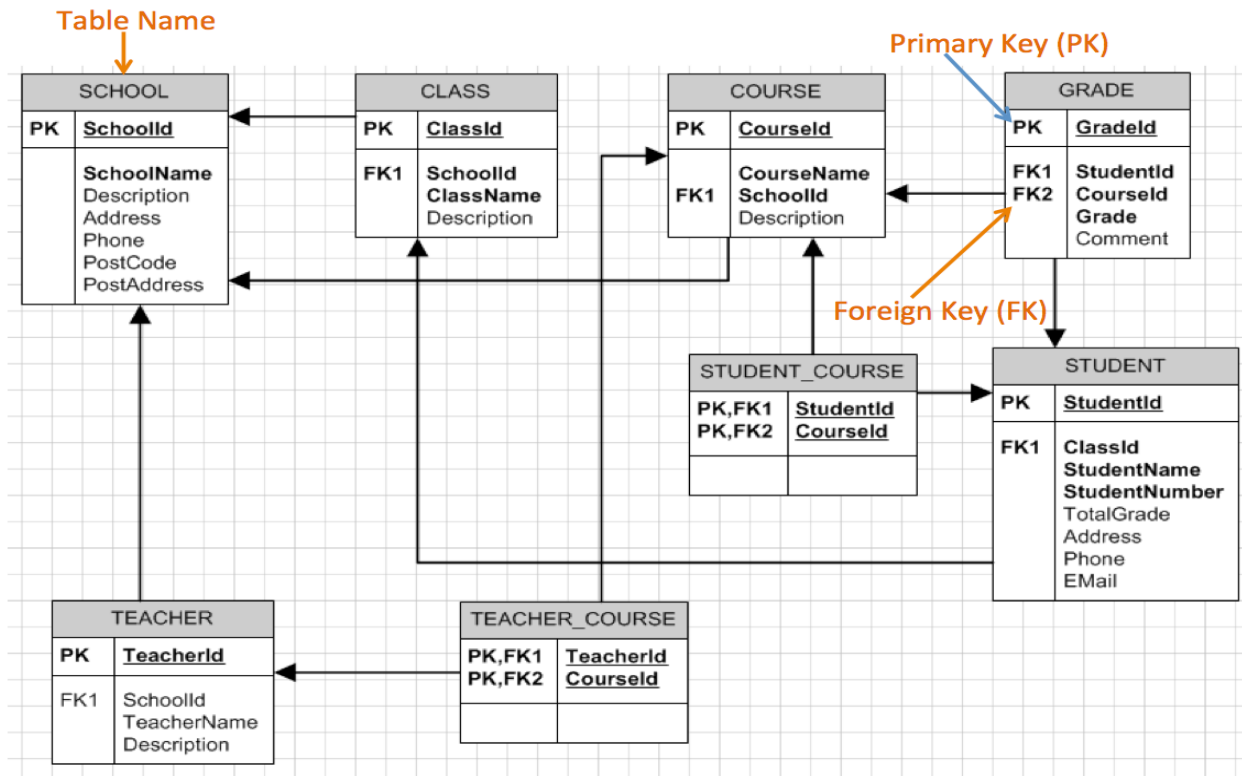


Figure 3-9: ER Diagram Example (MS Visio)

ERwin is a very good tool for creating ER diagrams, but it is expensive. But there exists a free edition called “CA ERwin Data Modeler Community Edition” – this is a free edition that contains a subset of the standard product.

In Figure 3-10 we see the same database model in ERwin as the example shown in Figure 3-9.

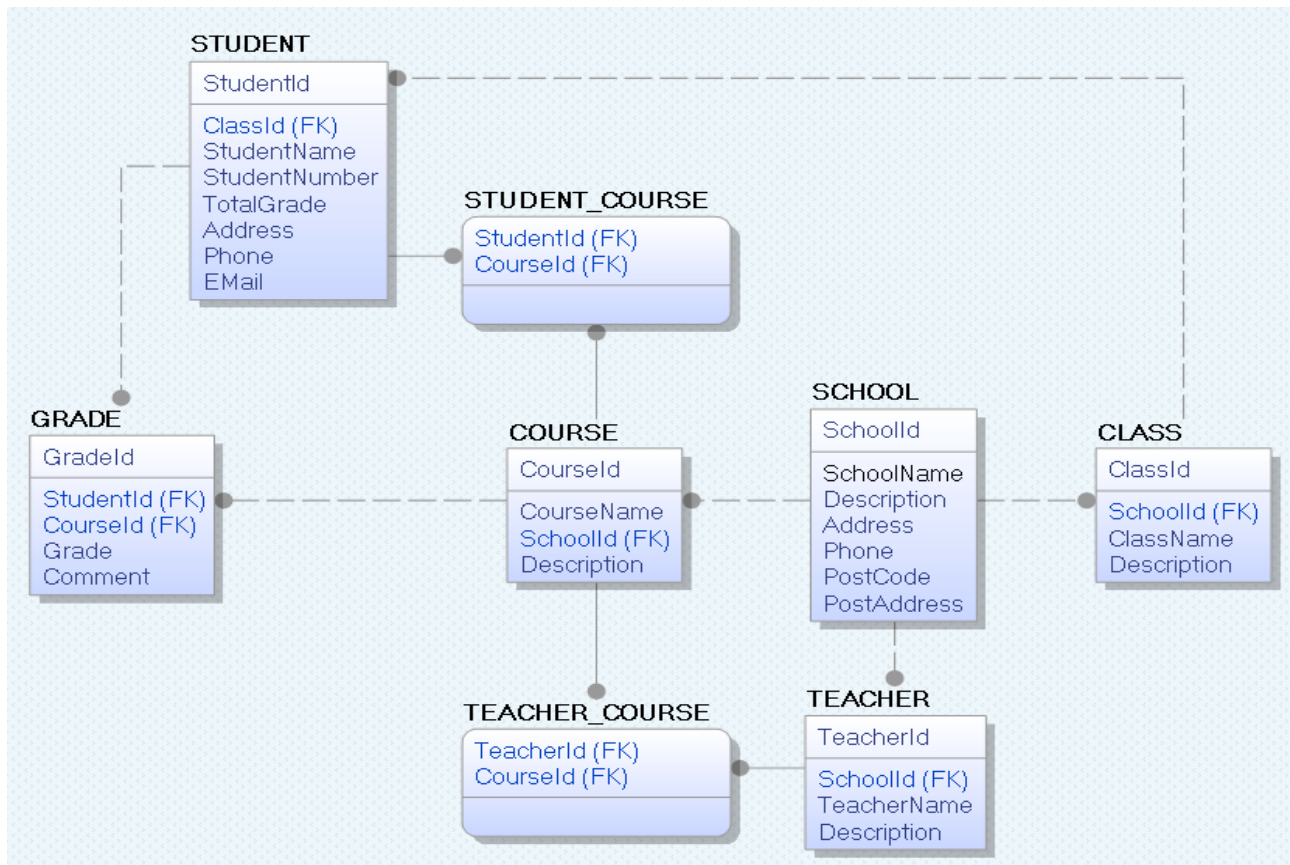


Figure 3-10: Database Modelling with ERwin

4 Visual Studio Team Services

Web: https://www.halvorsen.blog/documents/programming/software_engineering/vsts/

Visual Studio Team Services (VSTS), previously known as Visual Studio Online (VSO) is an Application Lifecycle Management (ALM) system, i.e., the system takes care of all aspects in software development from planning, requirements, coding, testing, deployment and maintenance.

Key Features:

- VSTS is a Source Code Control (**SCC**), Bug Tracking, Project Management, and Team Collaboration platform
- Integrated with Visual Studio
- VSTS in the Cloud (This means you don't need to host the server yourself)
- Free for up to 5 users

You don't need to install VSTS, but you need to create an account to get started (Figure 4-1):

www.visualstudio.com

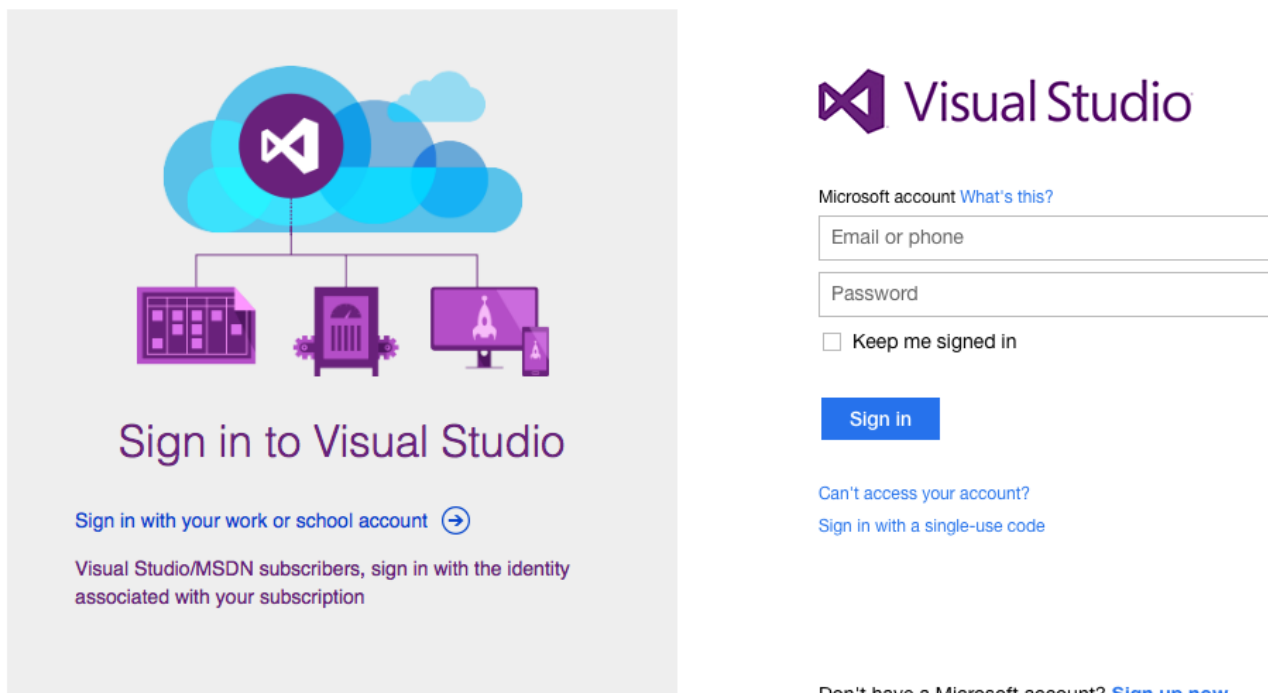


Figure 4-1: Visual Studio Team Services – Create an Account

Visual studio Team Services is a great tool if you work in a project where you need to share code, documents, etc. between the team members.

When you have created an account, you need to create a New Team Project (see Figure 4-2)

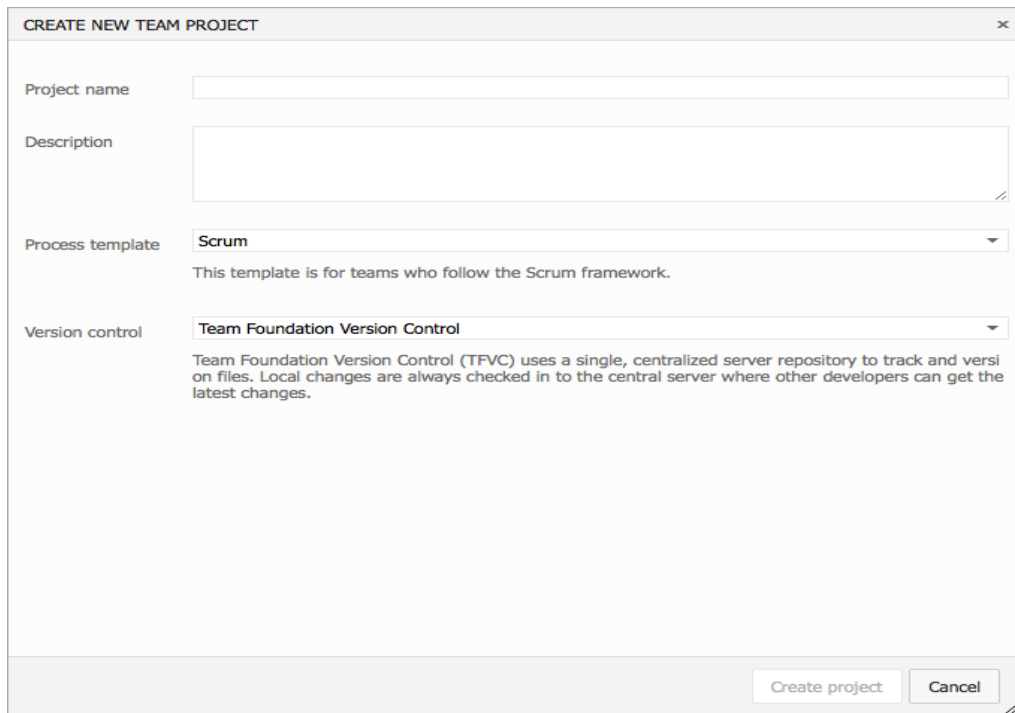
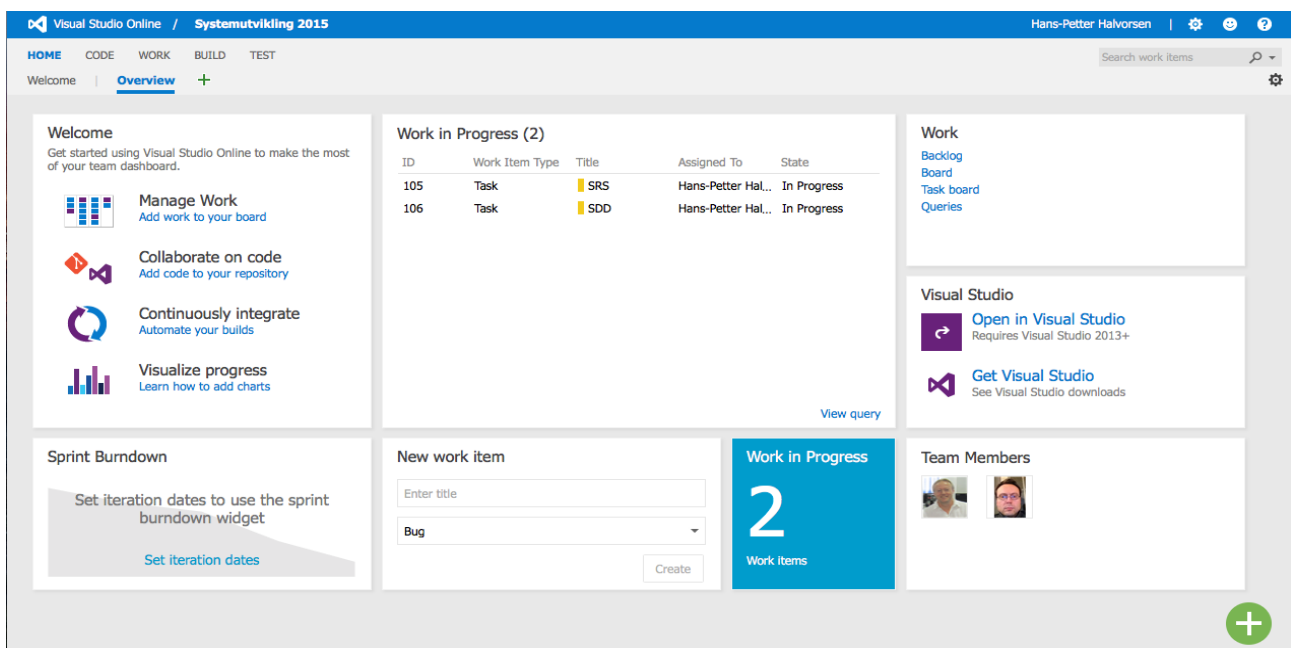


Figure 4-2: Create New Team Project

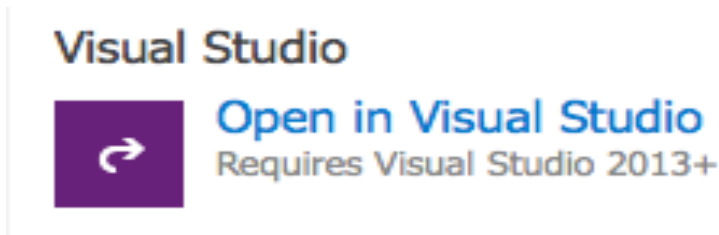
Then you can start to manage your Team Members, Make Plans and Requirements, Add Source Code, etc. See Figure 4-3.



ID	Work Item Type	Title	Assigned To	State
105	Task	SRS	Hans-Petter Hal...	In Progress
106	Task	SDD	Hans-Petter Hal...	In Progress

Figure 4-3: Visual Studio Team Services – Project Page

The coding is done in Visual Studio as normal, you just need to link your Visual Studio Team Services Team Project with Visual Studio. The easiest way to do this linking is to click “Open in Visual Studio”.



In Visual Studio you can add and upload code or other files to Visual Studio Online. This can be done directly from the Solutions Explorer (just right click and select Add Solution to Source Control...) or using the “Source Control Explorer”.

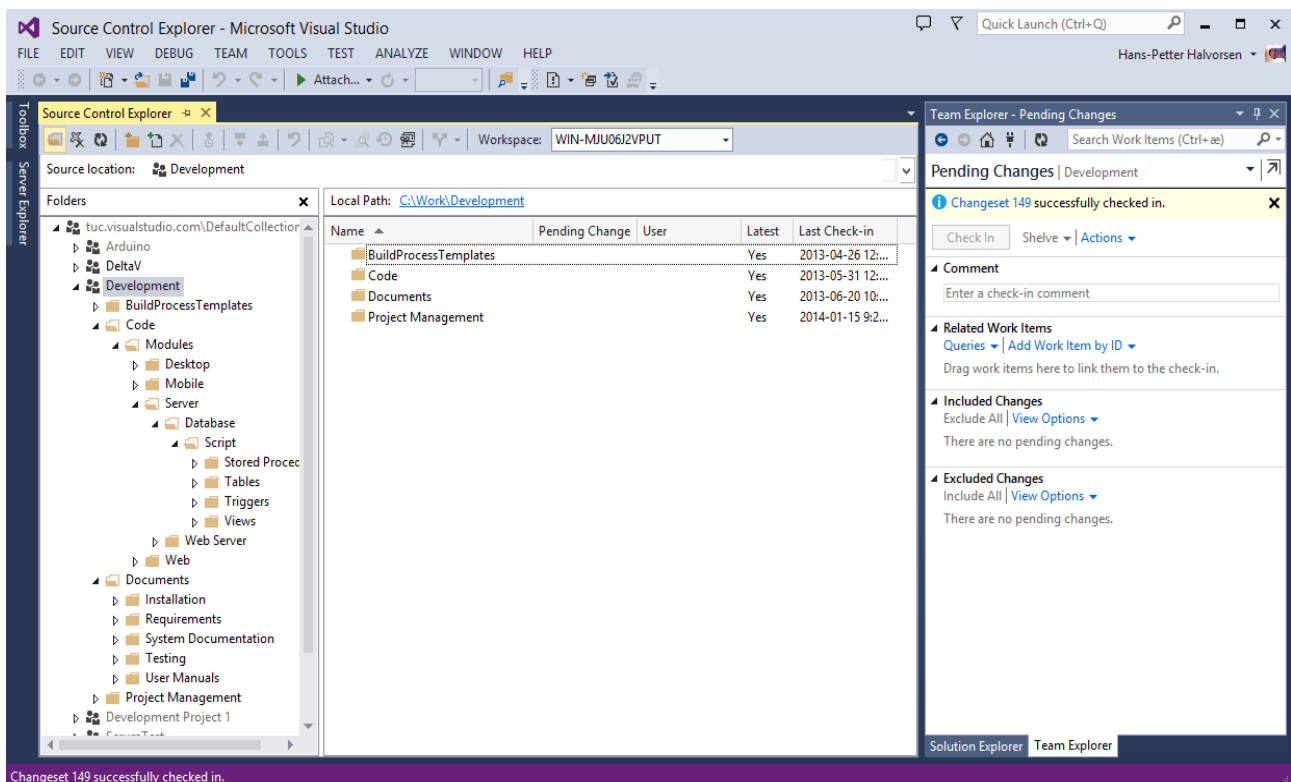


Figure 4-4: Using Visual Studio Team Services from Visual studio

If you are creating new projects, make sure to select “Add to source control” (Visual Studio Team Services) in the New Project dialog box. See Figure 4-5.

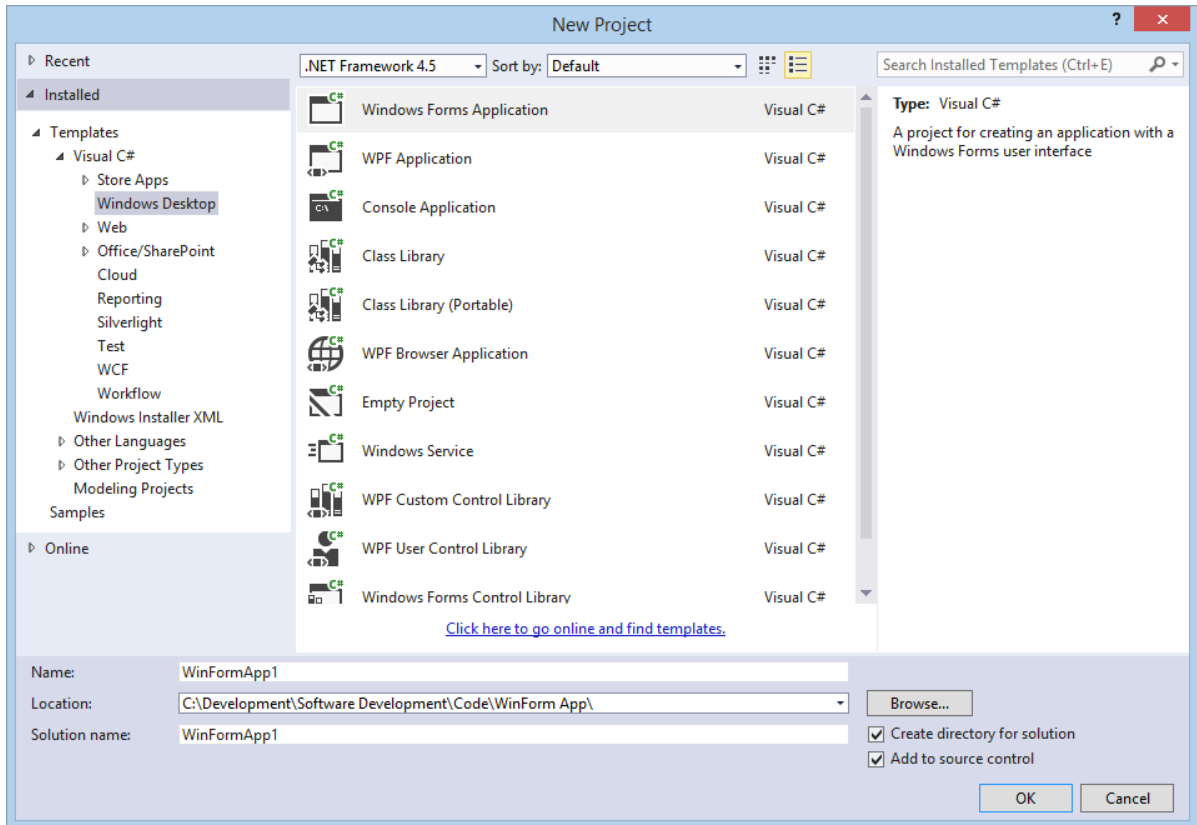


Figure 4-5: Add New Project to Source Control (VSTS)

5 Software Architecture

What is Software architecture? Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation (and Implementation) of these structures.

Figure 5-1 shows different types of software architecture.

Software Architecture Styles

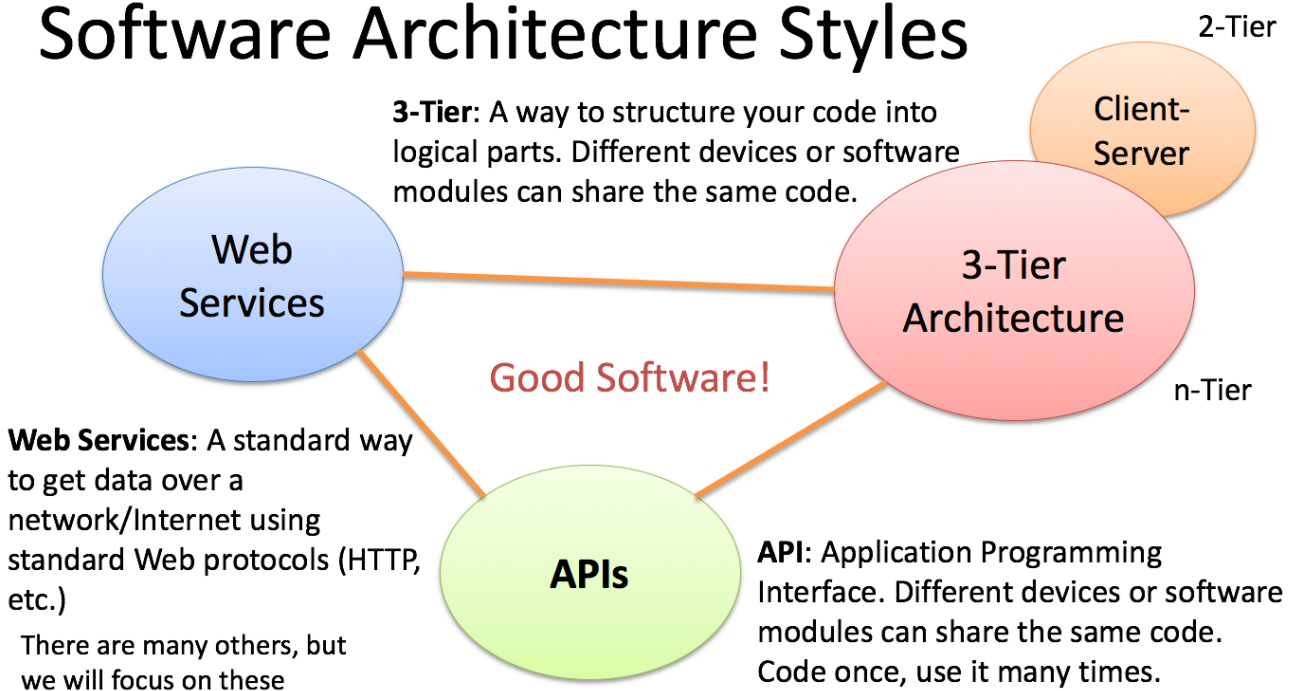


Figure 5-1: Software Architecture

We will discuss Web Services in detail in chapter 11 Web Services.

5.1 Client-Server Architecture

The Client-Server architecture is the most common type (see Figure 5-2).

OPC typically uses the Client-Server model, where we have an OPC Server and one or more OPC Clients. More about OPC in chapter 21 OPC.

Client-Server

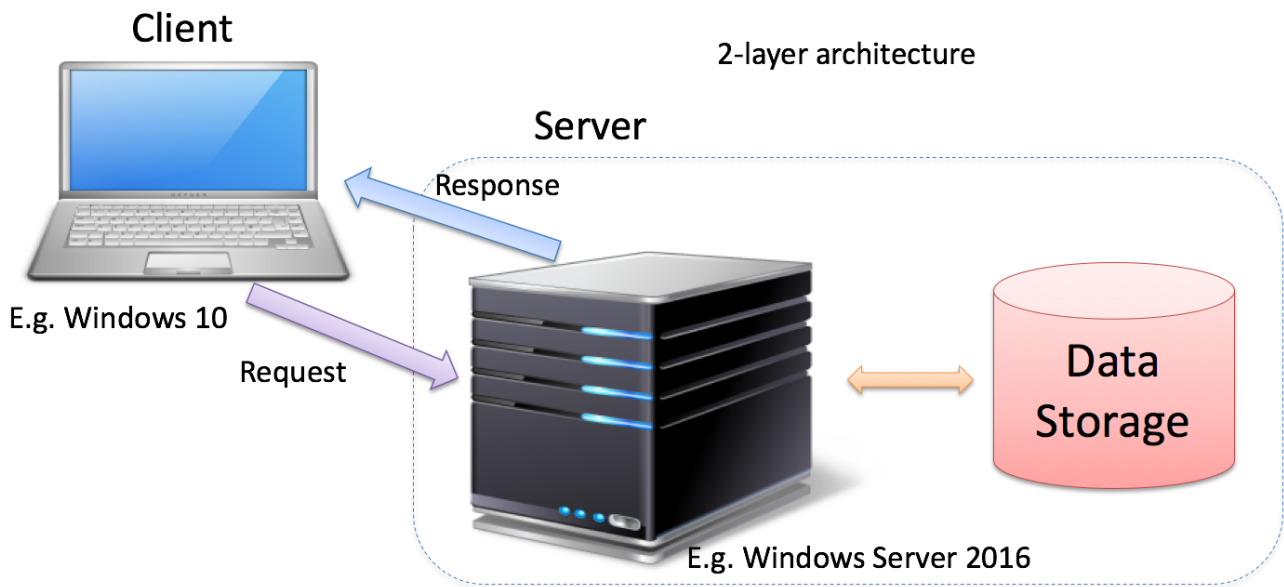


Figure 5-2: Client-Server Architecture

5.2 3-Tier Architecture

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client-server architecture in which presentation, application processing, and data management functions are physically separated. The most widespread use of multi-tier architecture is the three-tier architecture.

N-tier application architecture provides a model by which developers can create flexible and reusable applications.

By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application.

A three-tier architecture is typically composed of a presentation tier, a business/logic tier, and a data storage tier.

Figure 5-3 shows the 3-tier architecture model. This architecture is commonly used in modern software systems.

Note! The different layers can be on the same computer (Logic Layers) or on different Computers in a network (Physical Layers)

2-tier: The database-centric style. Typically, the clients communicate directly with the database.

A 3-tier style, in which clients do not connect directly to the database.

Why 3-Tier (N-Tier Architecture?). Here are some reasons:

- Flexible applications
- Reusable code
- Code once, use many times
- Modularized
- You need only to change part of the code
- You can deploy only one part
- You can Test only one part
- Multiple Developers
- Different parts (Tiers) can be stored on different computers
- Different Platforms and Languages can be used
- etc.

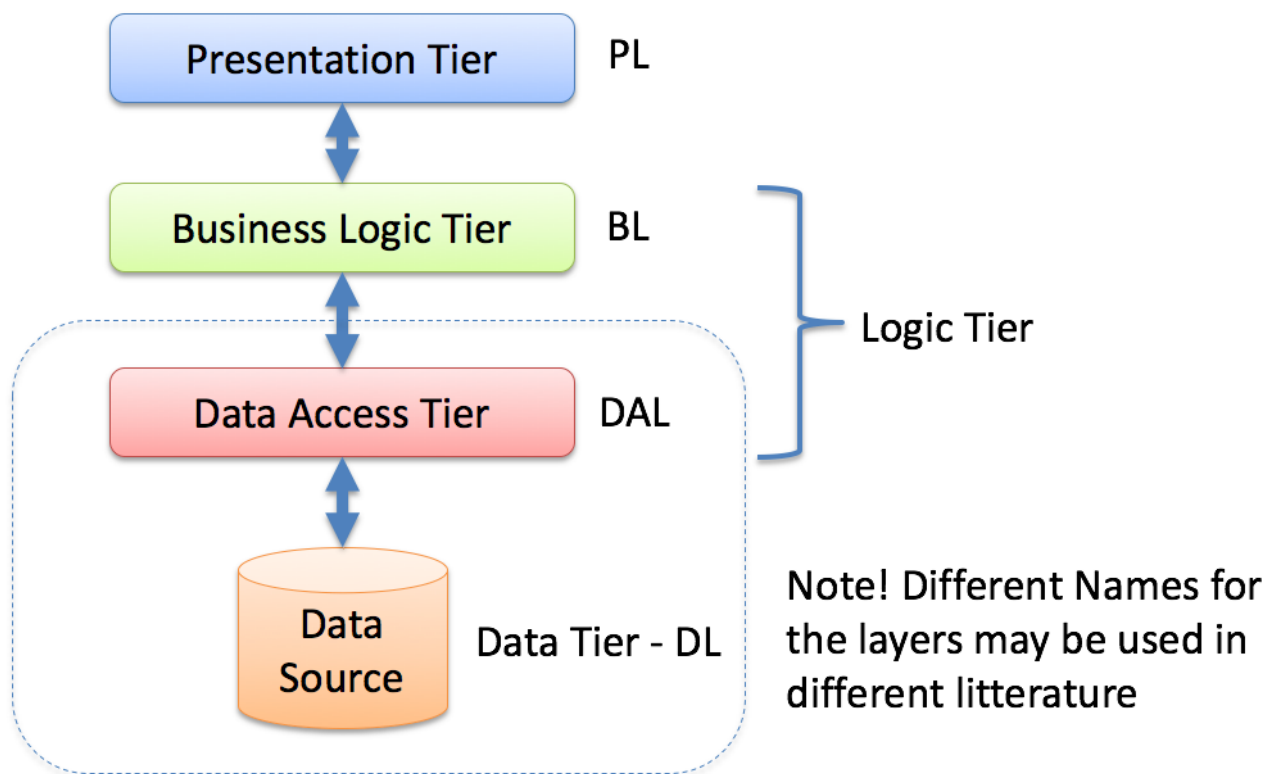


Figure 5-3: 3-Tier Architecture

Figure 5-4 shows a 3-tier architecture with physical layers.

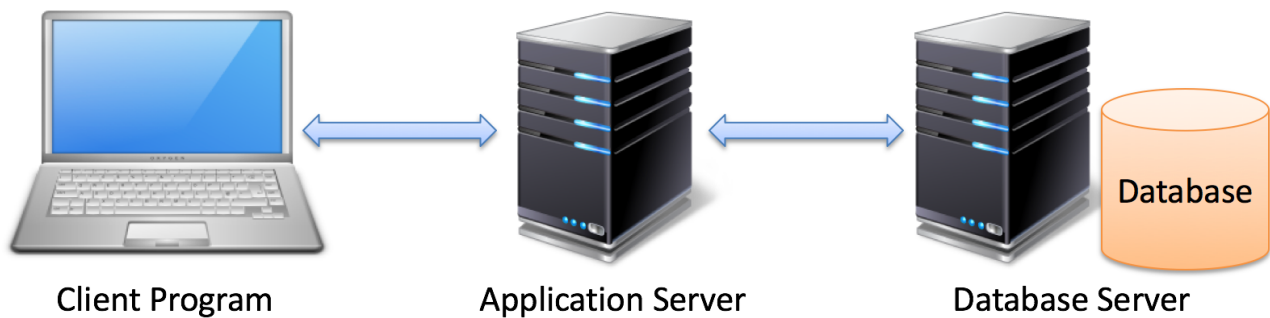


Figure 5-4: 3-Tier Application with Physical Layers

Presentation Tier

- This is the topmost level of the application.
- The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents.
- It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network.
- In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI

Application tier (business logic, logic tier, data access tier, or middle tier)

- The logical tier is pulled out from the presentation tier and, as its own layer.
- It controls an application's functionality by performing detailed processing.

Data tier

- This tier consists of database servers. Here information is stored and retrieved.
- This tier keeps data neutral and independent from application servers or business logic.
- Giving data its own tier also improves scalability and performance.

Figure 5-5 shows an example where an App for different platform, such as, e.g., PC, different types of smartphones and Tablets can share some of the same layers.

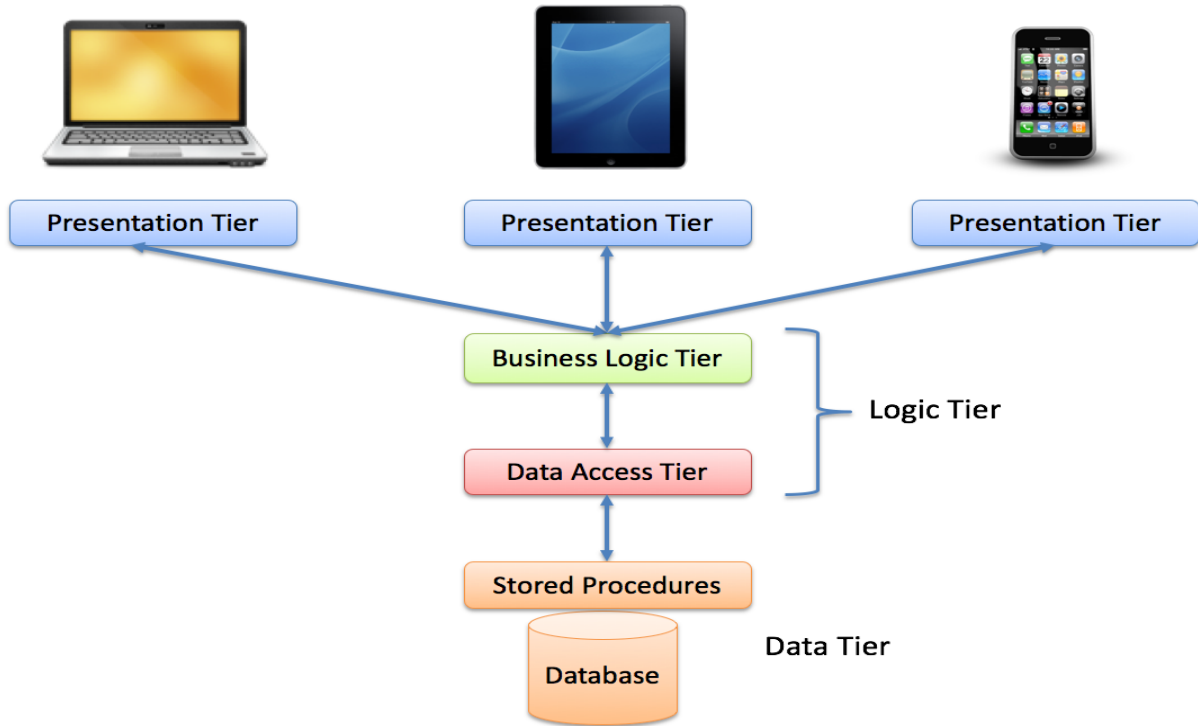


Figure 5-5: Different Devices sharing the same Layers

Typically, you can develop the server-side code which is common for all platforms. In addition you need to develop a Presentation tier for each platform (iOS, Android, Windows, ...).

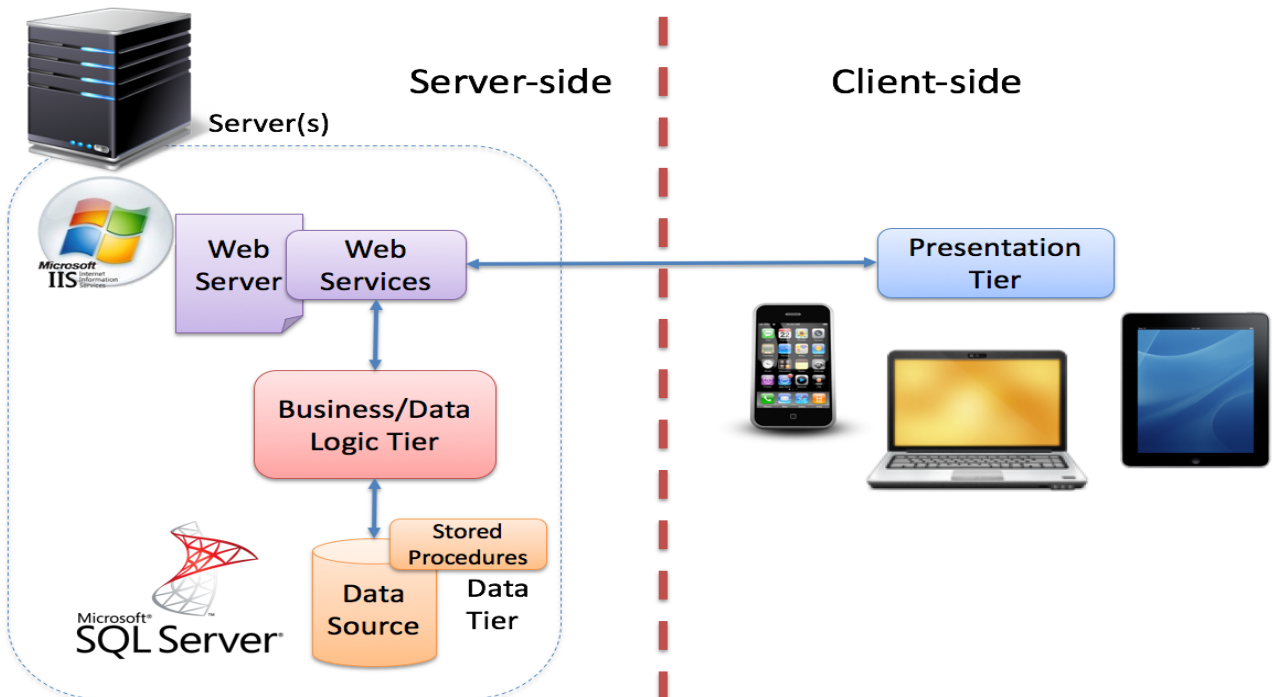
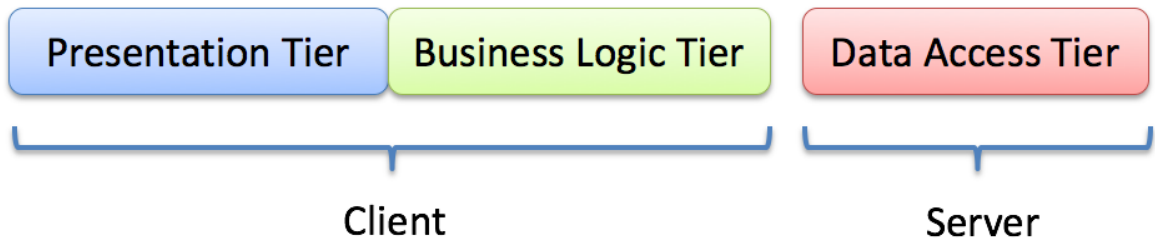


Figure 5-6: Server-side and Client-side Components

In Figure 5-7 we see how 3 layer architecture are used in a so-called Fat-Client vs. a Thin-Client. A Thin Client is typically a Web Page where all the Business and Data Logic runs on one or more servers.

Fat Client:



Thin Client:

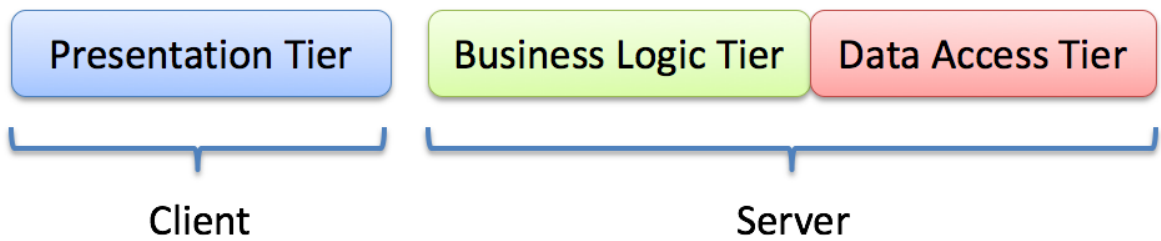


Figure 5-7: Fat Client vs. Thin Client

5.2.1 3-Tier Architecture with Visual Studio

Figure 5-8 shows how you can develop a 3-Tier (or a N-Tier/Multilayer) Application.

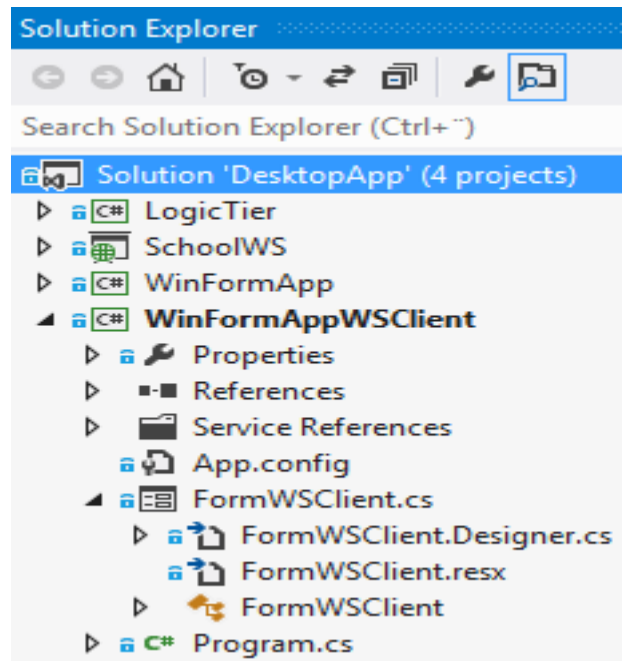


Figure 5-8: 3-Tier Architecture with Visual Studio

Each layer/tier are divided into Projects in Visual Studio. In the Solution explorer you can easily create new projects or add existing projects into a solution.

5.3 API

API - Application Programming Interface. A specification of how some software components should interact with each other. A library with functions, etc. you can use in your code

Examples:

- Windows API
- Java API
- ADO.NET
- Etc.

But you can (and you should!) also create your own API that you use internally in the team or expose to others.

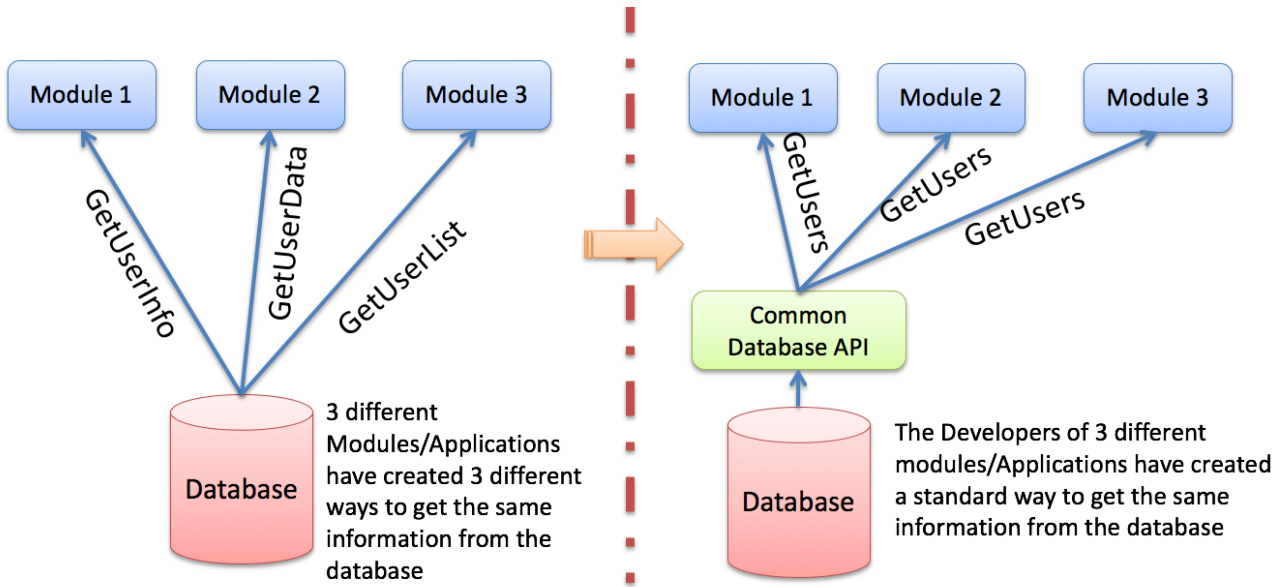


Figure 5-9: API Example

Figure 5-10 shows a typical example of development and use of a common Database API.

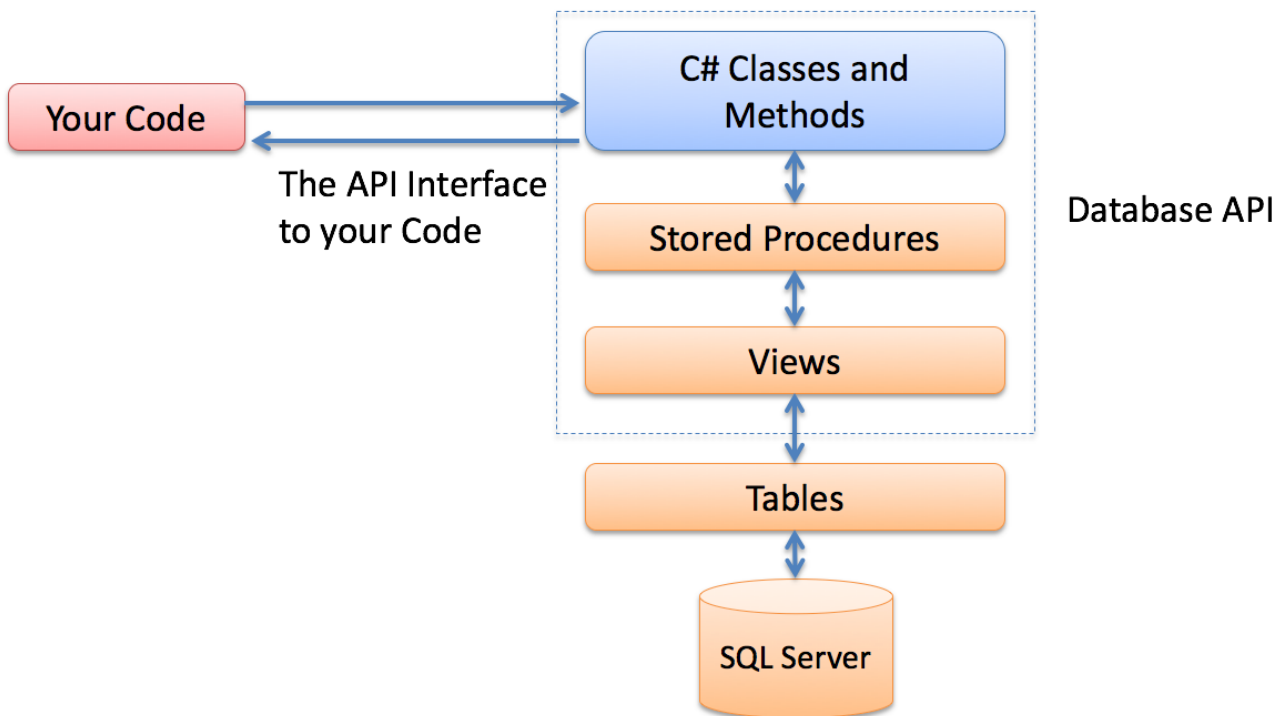


Figure 5-10: Typical Creation and Use of a Database API

Database Communication API. Stored Procedures and Views as a natural part of such a Database API. API using Layers: You A good practice is to create a New Project with one or more Classes/Methods where you put the API code that can be shared among the Developers. Each Developer can then add this Project to their Solution in order to use it and maintain it.

The benefits of API driven design:

When an API is used in a project, it

- Allows to focus on the project.
- Saves development time.
- Reduces errors and debugging.
- Facilitates modular design.
- Provides a consistent development platform.

API driven design requires planning and programming skills. API driven design is costly initially, but it pays in the long run. So, obviously, creating APIs is good software practice in most cases.

6 Implementation

The ultimate goal of most software engineering projects is to produce a working program.

The act of transforming the detailed design into a valid program in some programming language, together with all its supporting activities is referred to as implementation.

The implementation phase involves more than just writing code. Code also needs to be tested and debugged as well as compiled and built into a complete executable product.

We usually need to use a Source Code Control Tool in order to keep track of different versions of the code. Recommended tools are: Visual Studio Online or GitHub.

In many cases the detailed design is not done explicitly (in the Design Phase) but is left as part of the implementation. Doing the detailed design as part of the implementation is usually faster, but it may result in a less cohesive and less organized design, because the detailed design of each module will usually be done by a different person. In small projects, the detailed design is usually left as part of the implementation

In the practical implementation aspects of the topics in this document we will focus on the following programming languages:

- LabVIEW
- Visual Studio and C#
- MATLAB

These 3 tools are very different and all have their strengths and weaknesses. These tools also have different application areas.

6.1 LabVIEW

LabVIEW Is a graphical programming language used by scientists and engineers. LabVIEW is well suited for DAQ Applications and Control and Simulation Applications.

Since National Instruments (the vendor of LabVIEW) also make hardware and DAQ equipment, the integration with hardware and software are straightforward.

The vendor's web site: www.ni.com.

LabVIEW has the same things as other programming languages, but in a graphical way!

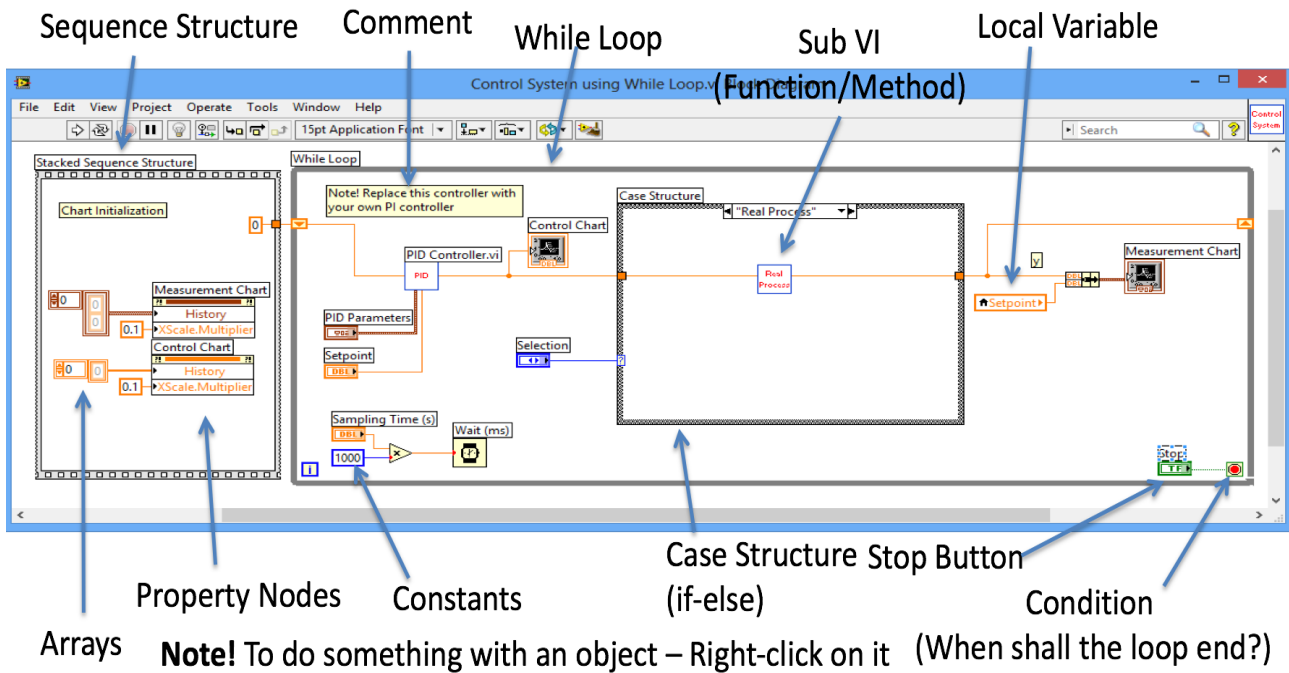


Figure 6-1: LabVIEW Example

LabVIEW Training Resources:

<https://www.halvorsen.blog/documents/programming/labview/>

6.2 Visual Studio and C#

C# is pronounced “see sharp”. C# is an object-oriented programming language and part of the .NET family from Microsoft. C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

C# is very similar to C++ and Java. C# is developed by Microsoft and works only on the Windows platform. C# is based on the .NET Framework (pronounced “dot net”).

.NET is a software framework that runs primarily on Microsoft Windows.

Visual Studio is the Integrated Development Environment (IDE) you use when programming in C# and the .NET platform.

The vendor's web site: www.visualstudio.com

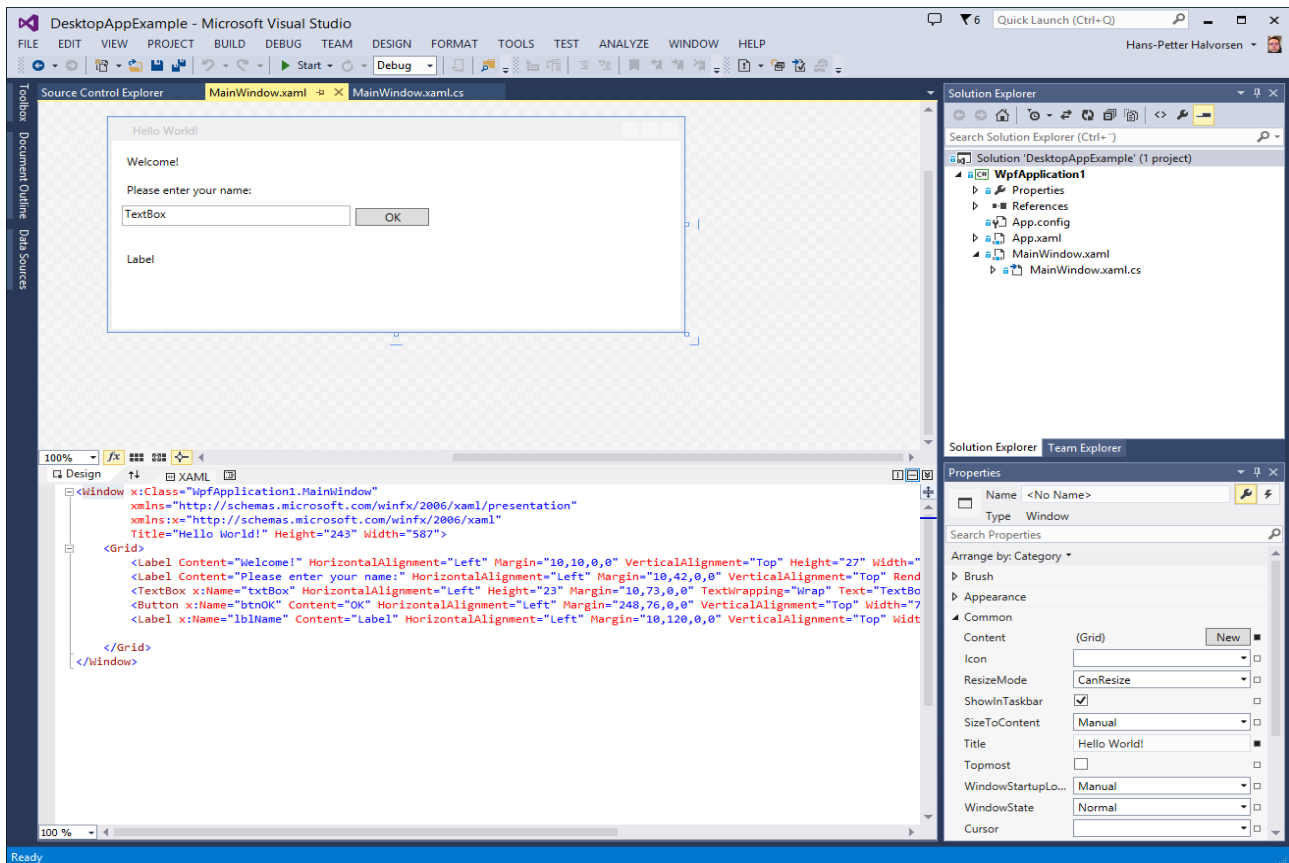


Figure 6-2: Visual Studio

6.3 MATLAB

MATLAB is a tool for technical computing, computation and visualization in an integrated environment. MATLAB is an abbreviation for MATrix LABoratory, so it is well suited for matrix manipulation and problem solving related to Linear Algebra, Modelling, Simulation and Control applications, etc.

MATLAB is very easy to use compared to similar tools, so it is a good tool to start with for inexperienced users. At the same time, it is very powerful and it can be used for advanced simulations, so you will never grow away from it - neither in school or work context.

The vendor's web site: www.mathworks.com

Figure 6-3 shows the MATLAB environment.

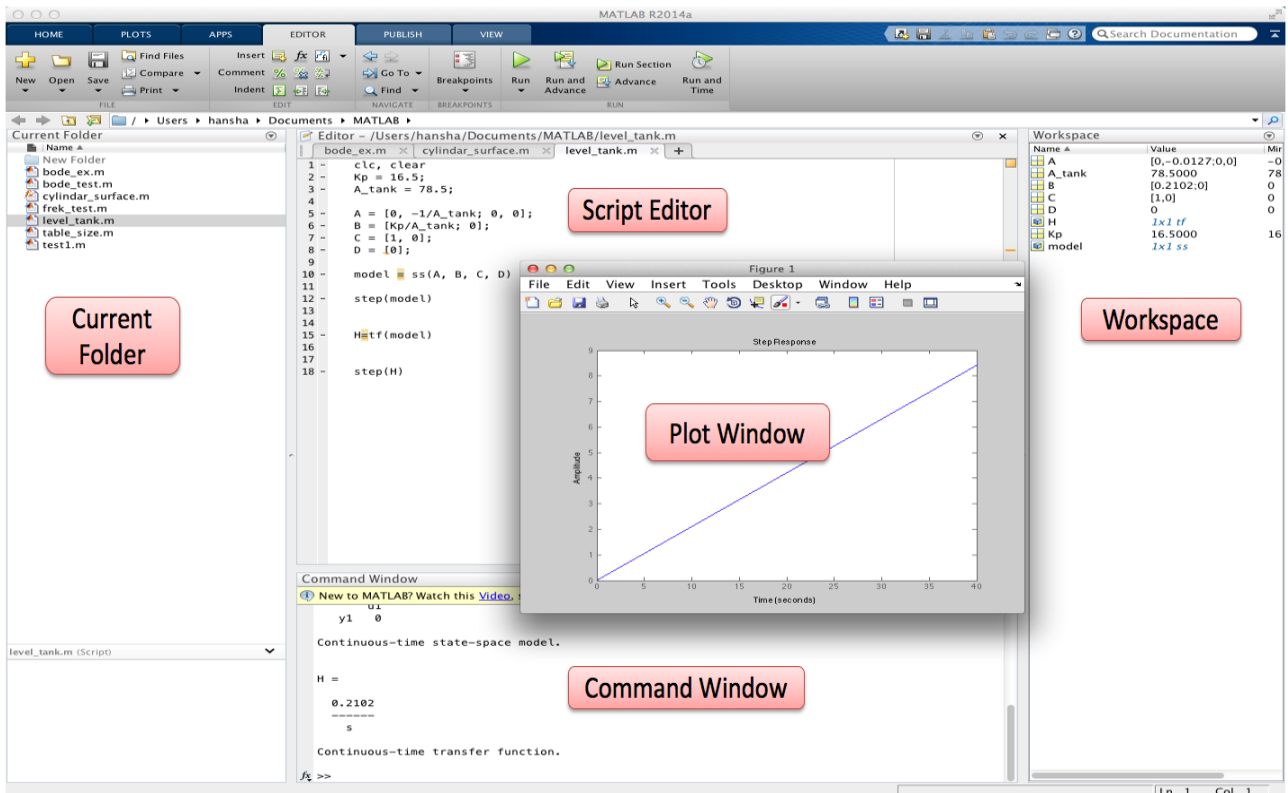


Figure 6-3: MATLAB

MATLAB Training Resources:

<https://www.halvorsen.blog/documents/programming/matlab/>

7 Testing

Web: https://www.halvorsen.blog/documents/programming/software_engineering/testing/

Different people have come up with various definitions for Software Testing, but generally, the goal with testing is:

- To ensure that the software meets the agreed requirements and design
- The application works as expected
- The application doesn't contain serious bugs
- Meets its intended use as per user expectations

Testing can be performed on different levels and by different persons. Testing is a very important part of software development. About 50% of the software development is about testing your software.

Since modern system engineering has become very complex, testing has become a very important part of any project.

Since Software Development today involves different platforms, different devices, network, servers and clients, etc., it has become very complex to test it. Today we have not only ordinary Desktop Apps, we have Web Apps, Mobile Apps, Apps for TVs, etc.

The software we create is typically a layer between the user of the software and the hardware and the operating system (Figure 7-1).

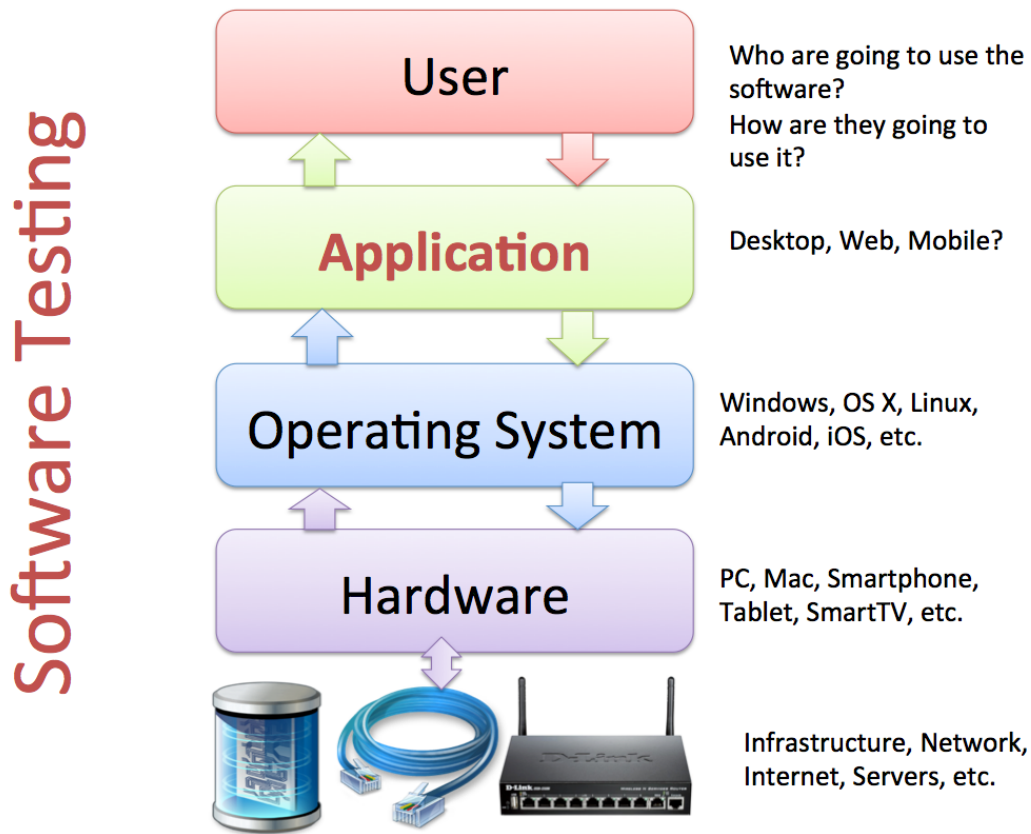


Figure 7-1: Components involved in Software Development & Testing

If we find bugs at the earlier stage, the cost to fix this will be less and thus it will reduce the overall cost of the application (Figure 7-2).

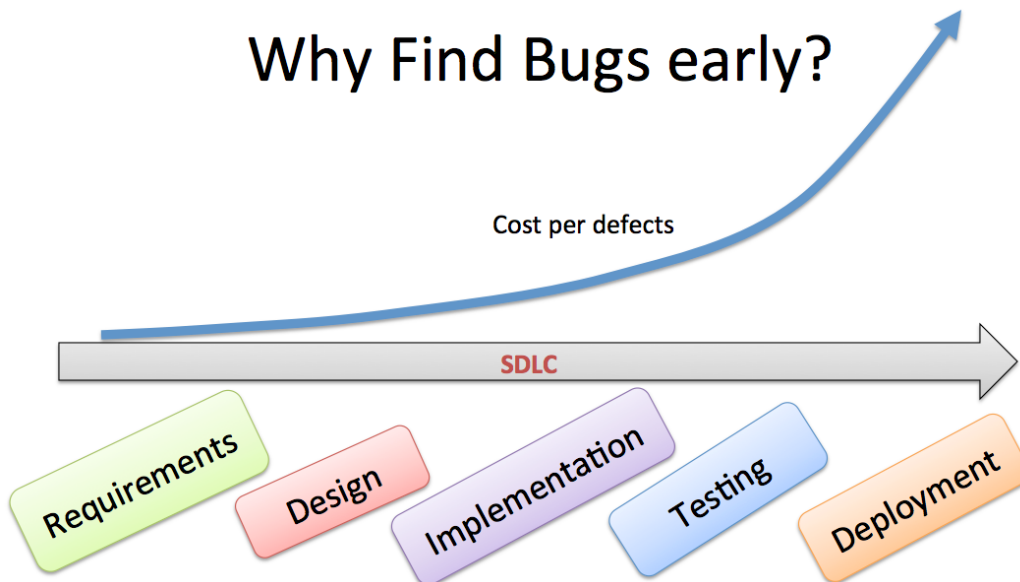


Figure 7-2: Find Bugs at an early stage

We have the following stages in testing:

1. **Development testing**, where the system is tested during development to discover bugs and defects. Development testing includes all testing activities that are carried out by the team developing the system.
2. **Release testing**, where a separate testing team test a complete version of the system before it is released to users.
3. **User testing**, where users or potential users of a system test the system in their own environment.

7.1 Levels of Testing

Since testing of advanced software systems is quite complex, we need a systematic approach to testing that involves different levels of testing (see Figure 7-3).

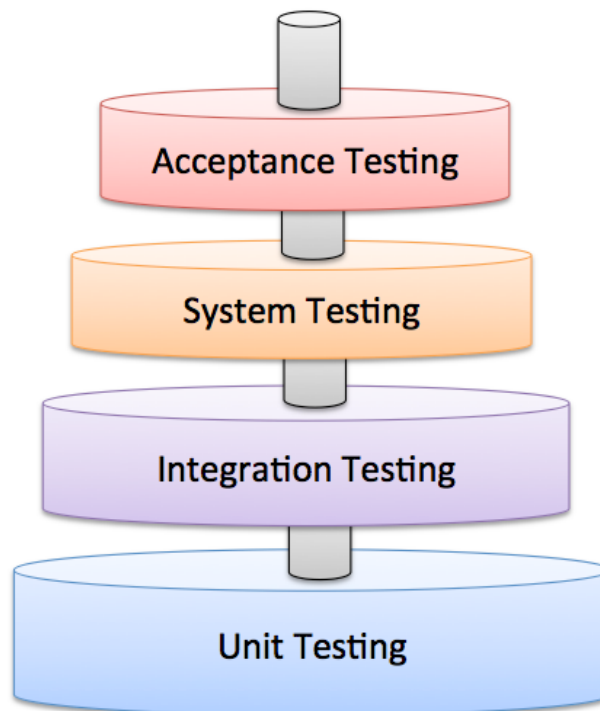


Figure 7-3: Systematic Testing

Short overview of the different Test levels in Figure 7-4 (more details later):

- Unit Tests are written by the Developers as part of the Programming. Each part is developed and Unit tested separately (Every Class and Method in the code)
 - Regression testing is testing the system to check that changes have not “broken” previously working code. Both Manually & Automatically (Re-run Unit Tests)
 - Integration testing means the system is put together and tested to make sure everything works together.
 - System or validation testing is Black-box Tests that validate the entire system against its requirements, i.e., Checking that a software system meets the specifications
-

- Acceptance Testing: The Customer needs to test and approve the software before he can take it into use. We have 2 types: FAT (Factory Acceptance Testing) and SAT (Site Acceptance Testing).

Levels of Testing

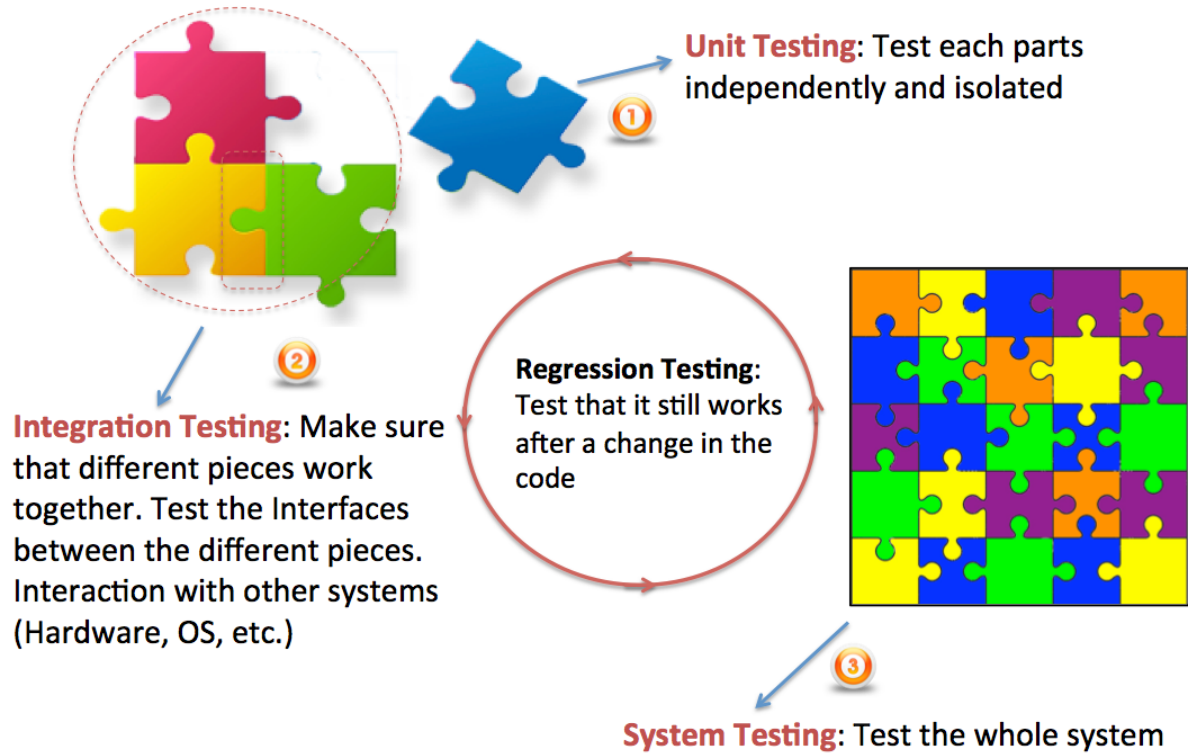


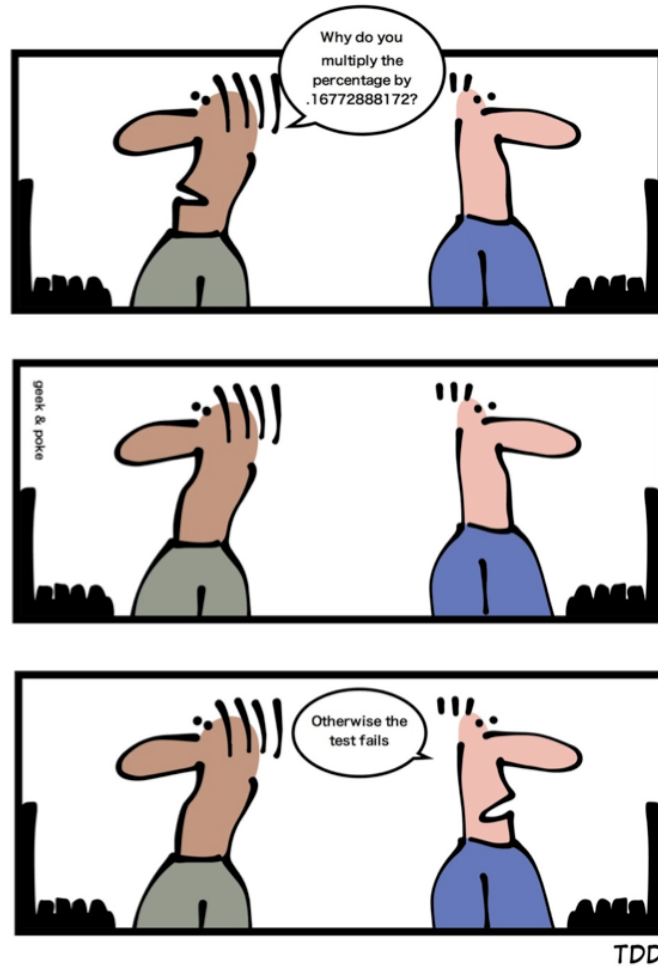
Figure 7-4: Levels of Testing

7.1.1 Unit Testing

Unit Testing (or *component testing*) refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class and methods level.

Unit Tests are written by the developers as part of the programming. They are automatically executed by the system, e.g., Visual Studio and Team Foundation Server have built-in functionality for Unit Testing.

Sometimes the Unit Tests are written before you start programming, so-called Test Driven Development (TDD).



<http://geek-and-poke.com>

Since Unit testing are part of the development process, so-called Unit Tests Framework are usually integrated with the IDE.

Unit Tests Frameworks:

- **Visual Studio Unit Test Framework.** Unit Tests are built into Visual Studio (no additional installation needed)
- JUnit (Java)
 - JUnit is a unit testing framework for the Java programming language.
- NUnit (.NET)
 - NUnit is an open source unit testing framework for Microsoft .NET. It serves the same purpose as JUnit does in the Java world
- PHPUnit (PHP)
- **LabVIEW Unit Test Framework Toolkit**

- etc.

Unit Testing in Visual Studio:

Visual Studio have built-in features for Unit Testing. In the Solution Explorer, you just add a “Test Project” as part of your code (see Figure 7-5).

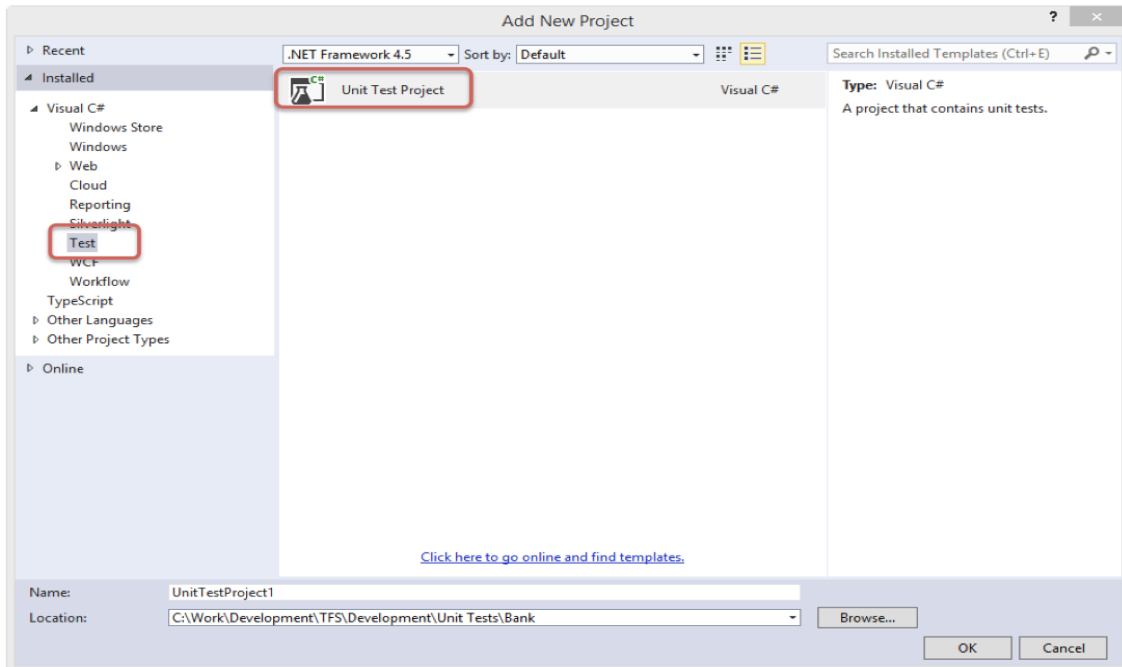


Figure 7-5: Unit Test Project in Visual Studio

In Figure 7-6 we see an example of how you create Unit Tests in Visual Studio and C#.

For Test Classes, you need to use [TestClass] and for Test Methods you need to use [TestMethod]. You also need to add a reference to the code under test (select “Add Reference” in the Solution Explorer and include “using <namespace>”) in your code.

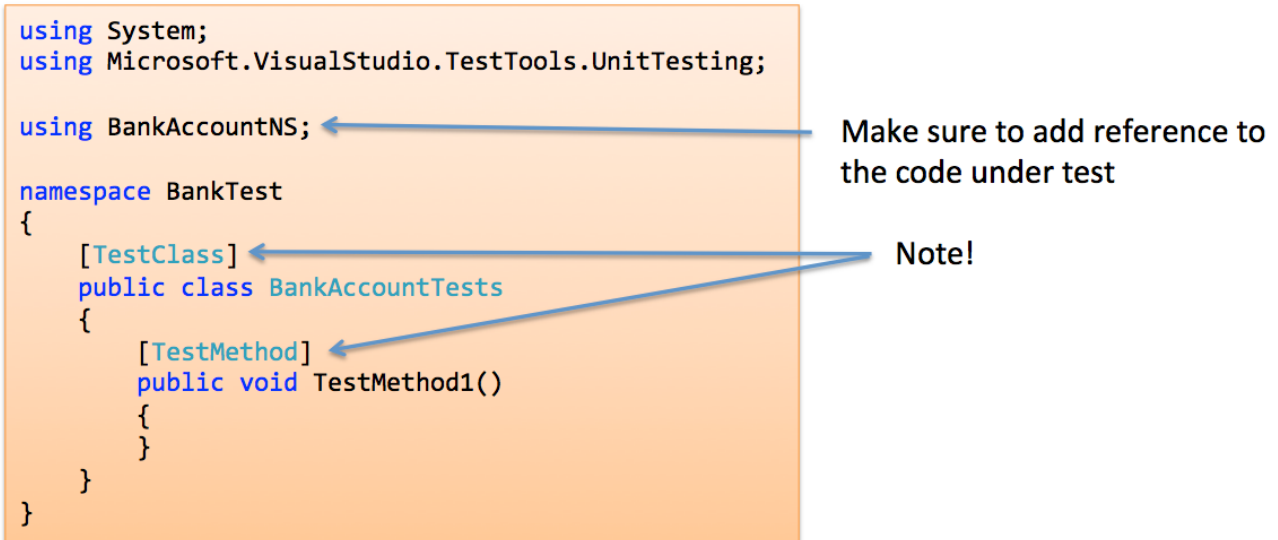


Figure 7-6: Unit Test Principle in Visual Studio and C#

The basic concept in Unit Testing is to Compare the results when running the Methods with some Input Data (“Actual”) with some Known Results (“Expected”)

Example:

```
Assert.AreEqual(expected, actual, 0.001, "Test failed because...");
```

Unit Tests – Best Practice:

- A Unit Test must only do one thing
- Unit Test must run independently
- Unit Tests must not depend on the environment
- Test Functionality rather than implementation
- Test public behavior; private behavior relates to implementation details
- Avoid testing UI components
- Unit Tests must be easy to read and understand
- Create rules that make sure you need to run Unit Tests (and they need to pass) before you are allowed to Check-in your Code in the Source Code Control System

7.1.2 Regression Testing

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, or old bugs that have come back.

1. Regression testing is testing the system to check that changes have not “broken” previously working code.
2. In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.
3. Tests must run “successfully” before the change is committed.

7.1.3 Integration Testing

Integration testing verifies the interfaces between components against a software design.

7.1.4 System Testing/Validation Testing

System Testing follows Integration Testing. It consists of Black-box Tests that validate the entire system against its requirements. System Testing is about checking that a software system meets specifications and that it fulfills its intended purpose. System Testing is often executed by an independent group (QA group). QA – Quality Assurance.

Since system tests make sure the requirements are fulfilled, they must systematically validate each requirement in the SRS (Software Requirements Specification).

7.1.5 Acceptance Testing

Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. It is primarily for custom systems.

In Figure 7-7 we see a typical acceptance test process.

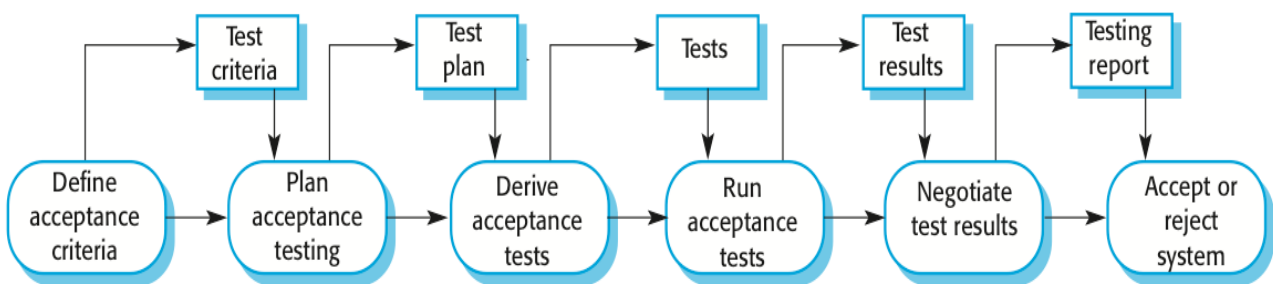


Figure 7-7: Acceptance Testing

The steps are:

- Define acceptance criteria
- Plan acceptance testing
- Derive acceptance tests

- Run acceptance tests
- Negotiate test results
- Reject/accept system

We have 2 main types of Acceptance Testing:

- FAT – Factory Acceptance Testing
- SAT – Site Acceptance Testing

FAT – Factory Acceptance Testing is usually performed in the Test Environment at the software company.

SAT – Site Acceptance Testing is performed at the Customer in the actual Production Environment. This is the final step to determine if the requirements of a specification or contract are met.

If the test is accepted, the software is officially handed over to the customer.

Note! Other terms and definitions are used as well in different literature.

7.2 Test Categories

We can divide testing into 2 different categories, which is:

- Black-box Testing
- White-box Testing

7.2.1 Black-box Testing

Black-box testing is a method of software testing that examines the functionality of an application (what the software does) without going inside the internal structure (White-box Testing).

You need no knowledge of how the system is created. Black-box testing can be done by a person who only know what the software is supposed to do. You may compare to driving a Car – you don't need to know how it is built in order to test it.

7.2.2 White-box Testing

In White-box Testing you need to have knowledge of how (Design and Implementation) the system is built. White-box Testing is also called “Glass-box testing”.

In Figure 7-8 we see how White-box Testing is working.

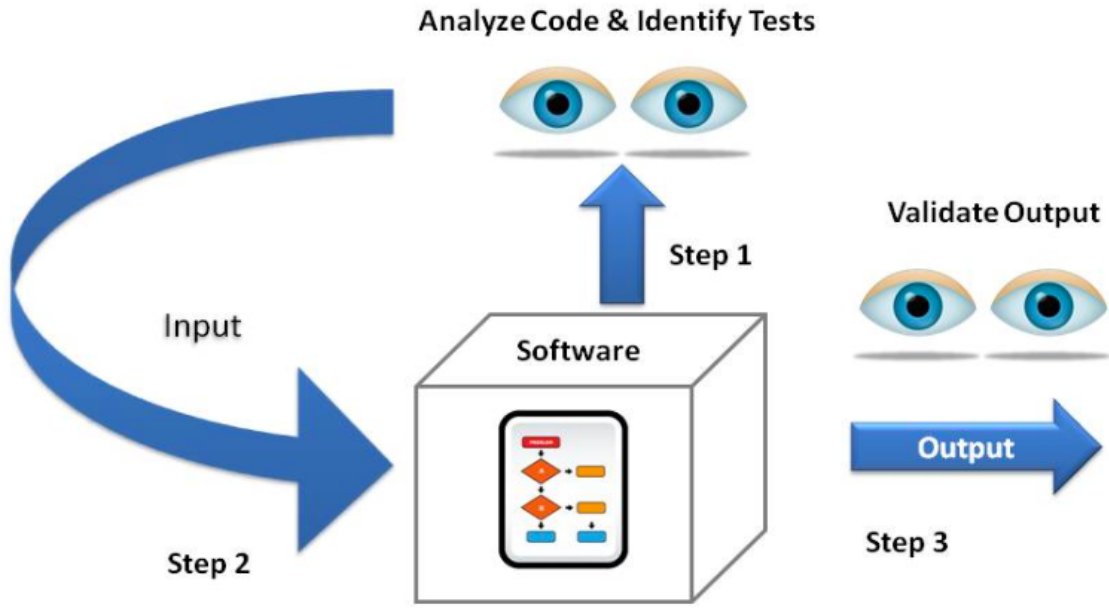


Figure 7-8: White-box Testing

8 Documentation

During the software development, a lot of documentation (Figure 8-1) is created in the different phases of the development.

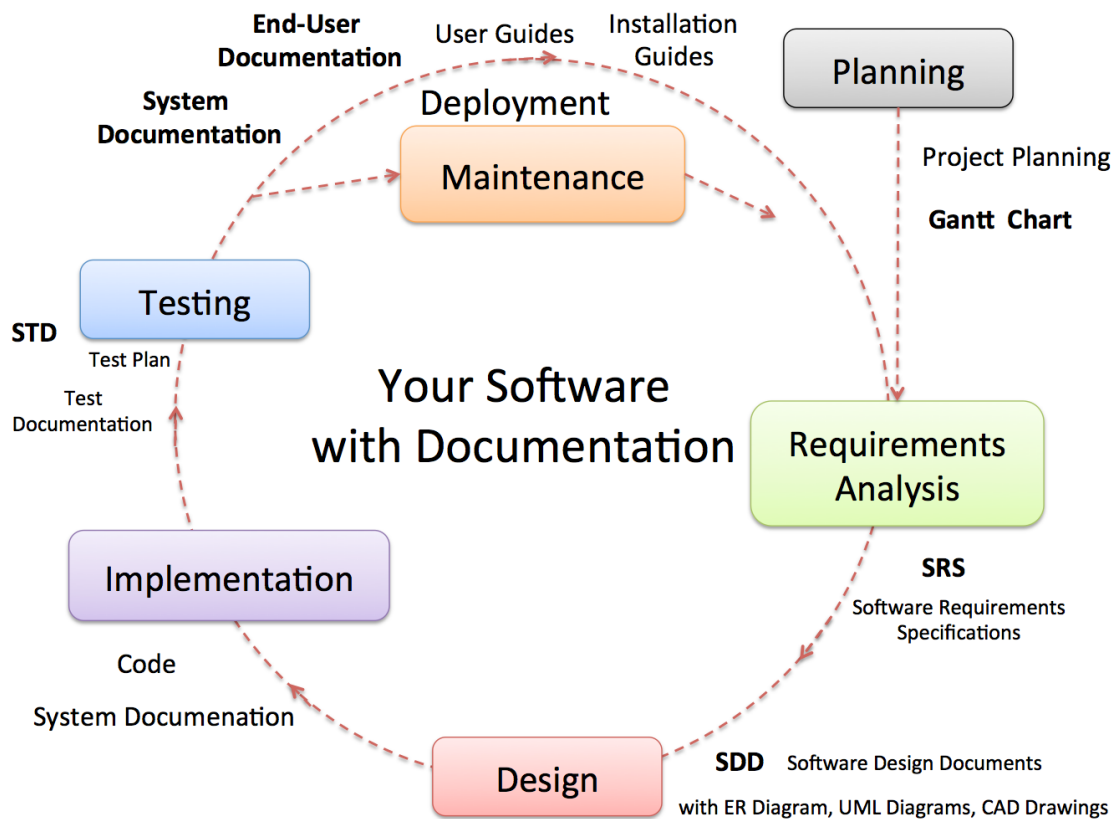


Figure 8-1: Example of Documentation during the SDLC

Some documents are for internal use inside the software company or inside the development team, while other documents are important for the stakeholders and the customers that are going to use the software (Figure 8-2).

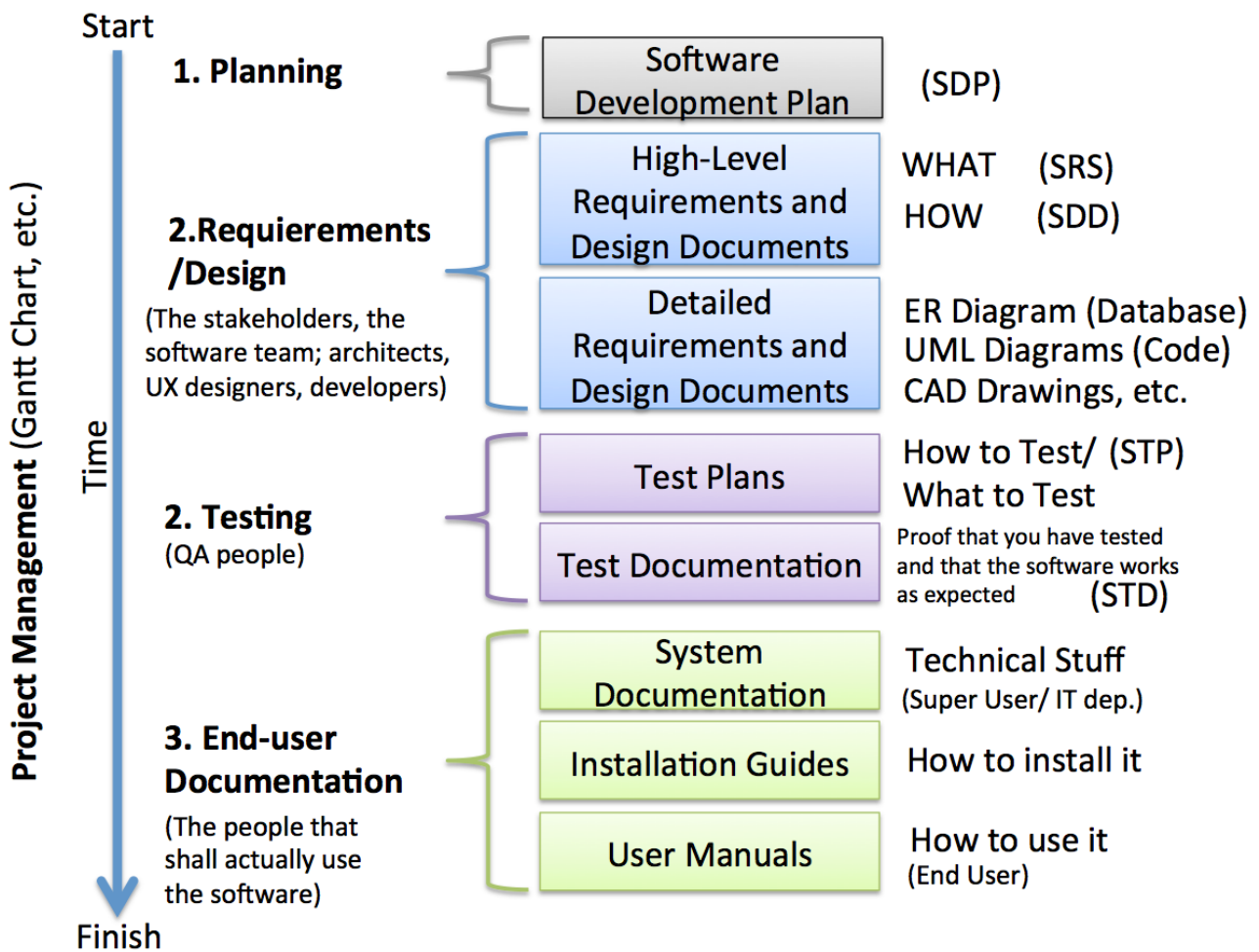


Figure 8-2: Software Documentation

Some important documents are:

- **SDP** – Software Development Plan
- **SRS** – Software Requirements Specifications
 - A document stating what an application must accomplish
- **SDD** – Software Design Document
 - A document describing the design of a software application
- **STP** - Software Test Plan
 - Documentation stating what parts of an application will be tested, and the schedule of when the testing is to be performed
- **STD** - Software Test Documentation
 - Introduction, Test Plan, Test Design, Test Cases, Test procedures, Test Log, ..., Summary

See Figure 8-3 for an overview of documentation categories used in a project.

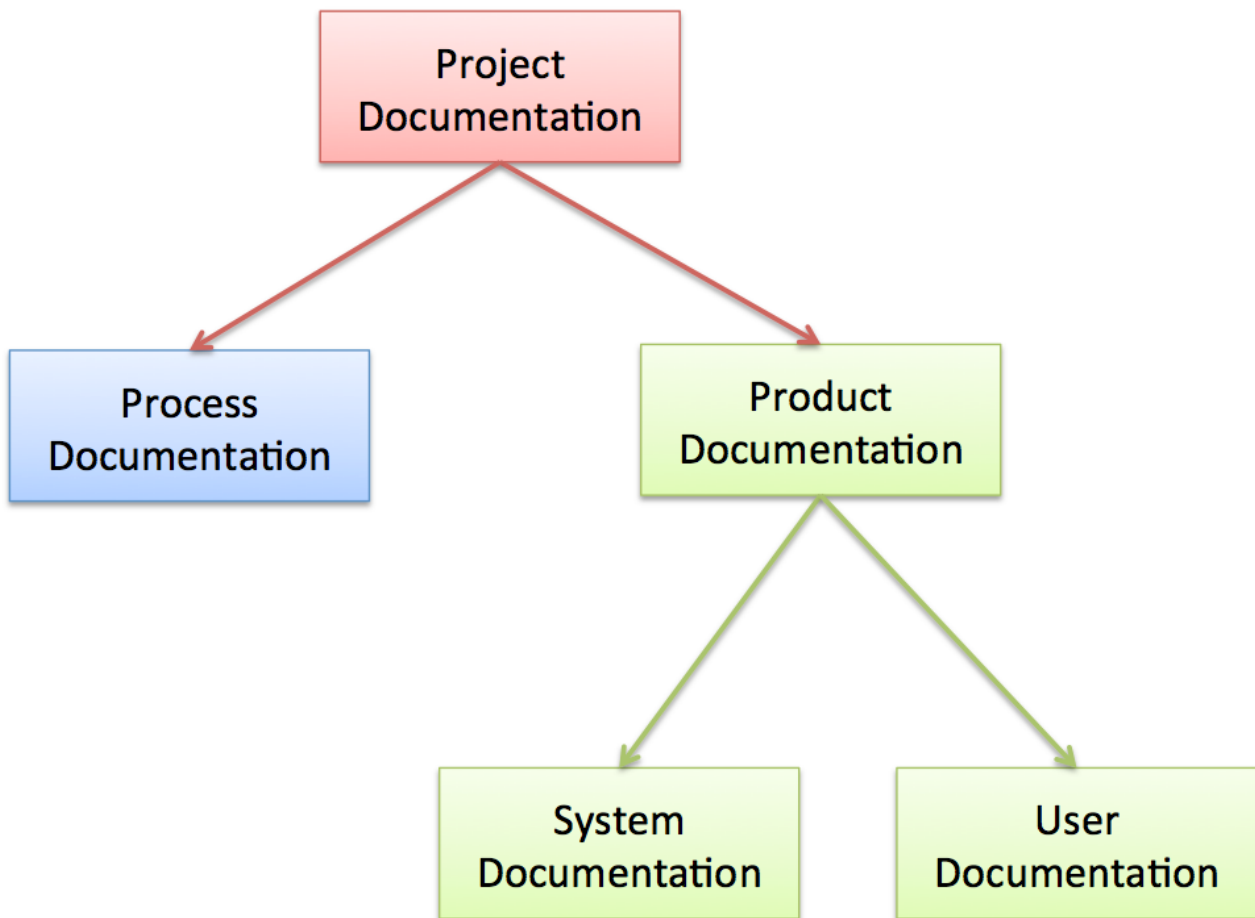


Figure 8-3: Software Project Documentation

Documentation produced during a software Project can be divided into 2 main categories:

- **Process Documentation**
 - These documents record the process of development and maintenance, e.g., Plans, Schedules (e.g., Gantt Charts), etc.
- **Product Documentation**
 - These documents describe the product that is being developed. Can be divided into 2 sub categories:
 - **System Documentation**
 - Used by engineers developing and maintaining the system
 - **User Documentation**
 - Used by the people that is using the system

Here are some Software Documentation Requirements:

- Should act as a communication medium between members of the Development Team (Process Documentation)
- Information repository used by Maintenance Engineers (Product Documentation)
- Information for Management to help them Plan, Budget and Schedule the Software Development Process (Process Documentation)
- Some of the documents should tell users how to use and administer the system (Product Documentation)
- Documents for Quality Control, System Certification, etc. (Process/Product Documentation)

Satisfying these requirements requires different types of documents from informal working documents through professionally produced User Manuals

8.1 Process Documentation

Purpose:

1. Process Documentation is produced so that the development of the system can be managed
2. It is an essential component of plan-driven approaches (e.g., Waterfall)
3. Agile Approaches: The Goal is to minimize the amount of Process Documentation

We have different categories of Process Documentation:

- **Plans, estimates and schedules.** These are documents produced by managers which are used to predict and to control the software process.
 - **Reports.** These are documents which report how resources were used during the process of development.
 - **Standards.** These are documents which set out how the process is to be implemented. These may be developed from organizational, national or international standards.
 - **Working papers.** These are often the principal technical communication documents in a project. They record the ideas and thoughts of the engineers working on the project, are interim versions of product documentation, describe implementation strategies and set out problems which have been identified. They often, implicitly, record the rationale for design decisions.
-

- **E-mail messages, wikis, etc.** These records the details of everyday communications between managers and development engineers.

8.2 Product Documentation

Purpose:

- Describing the delivered software product
- Unlike most process documentation, it has a relatively long life. It must
- Evolve in step with the product that it describes. Product documentation includes
 - User documentation, which tells users how to use the software product,
 - System Documentation, which is principally intended for maintenance engineers.

8.2.1 System Documentation

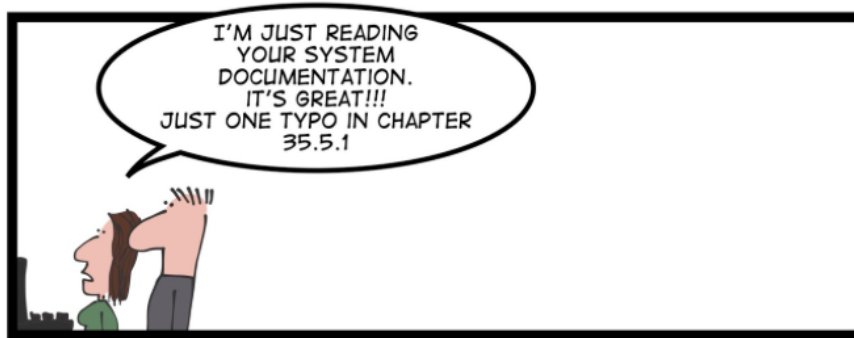
The system documentation describes how the system is designed and how it works in detail.

1. System documentation includes all of the documents describing the system itself from the requirements specification to the final acceptance test plan.
2. Documents describing the design, implementation and testing of a system are essential if the program is to be understood and maintained.
3. Like user documentation, it is important that system documentation is structured, with overviews leading the reader into more formal and detailed descriptions of each aspect of the system.

For large systems that are developed to a customer's specification, the system documentation should include:

- The **requirements** document.
 - A document describing the **system architecture**.
 - For each program in the system, a description of the architecture of that program.
 - For each component in the system, a description of its functionality and interfaces.
 - Program **source code** listings, which should be commented where the comments should explain complex sections of code and provide a rationale for the coding method used.
-

- If meaningful names are used and a good, structured programming style is used, much of the code should be self-documenting without the need for additional comments.
- This information is now normally maintained electronically rather than on paper with selected information printed on demand from readers.
- **Validation** documents describing how each program is validated and how the validation information relates to the requirements.
 - These may be required for the quality assurance processes in the organization.
- **A System Maintenance Guide**, which describes known problems with the system, describes which parts of the system are hardware and software dependent and which describes how evolution of the system has been taken into account in its design



BE AWARE!!!



SOMEBODY MAY ACTUALLY READ IT!

8.2.2 User Documentation

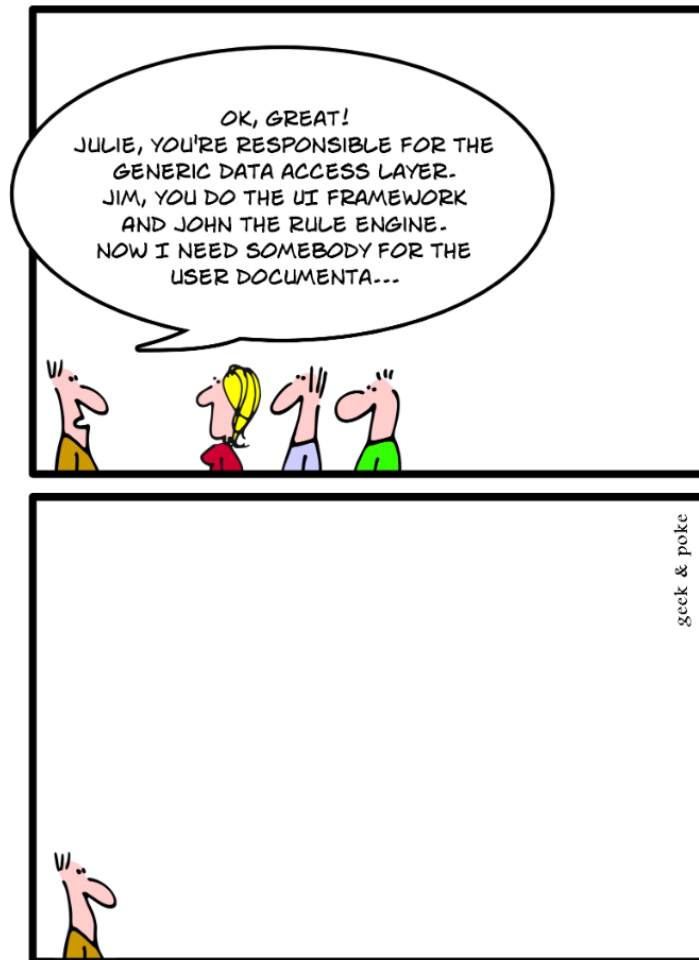
Users of a system are not all the same. The producer of documentation must structure it to cater for different user tasks and different levels of expertise and experience.

It is particularly important to distinguish between end-users and system administrators:

- **End-users** use the software to assist with some task.
 - This may be flying an aircraft, managing insurance policies, writing a book, etc. They want to know how the software can help them. They are not interested in computer or administration details.
- **System administrators** are responsible for managing the software used by end-users.
 - This may involve acting as an operator if the system is a large mainframe system, as a network manager if the system involves a network of workstations or as a technical guru who fixes end-users software problems and who liaises between users and the software supplier.

We have different user documentation, such as:

- User Manual
 - Installation Guide
 - Wiki
 - etc.
-



<http://geek-and-poke.com>

User Manual:

A user guide or user's guide, also commonly known as a manual, is a technical communication document intended to give assistance to people using a particular system. It is usually written by a technical writer, although user guides are written by programmers, product or project managers, or other technical staff, particularly in smaller companies

The sections of a user manual often include:

- A cover page
- A title page and copyright page
- A preface, containing details of related documents and information on how to navigate the user guide
- A content page
- **A guide on how to use at least the main functions of the system (Text + Screen Shots)**

- A troubleshooting section detailing possible errors or problems that may occur, along with how to fix them
 - A FAQ (Frequently Asked Questions)
 - Where to find further help, and contact details
 - A glossary and, for larger documents, an index
-

Part 2 : Industrial IT

In this part, we introduce important topics within Industrial IT, such as Data Communication, Database Systems, Web Services, Modbus, Virtualization, Wireless Systems.

9 Data Communication

Computer hardware and software require each other and neither can be realistically used without the other, see Figure 2-1.

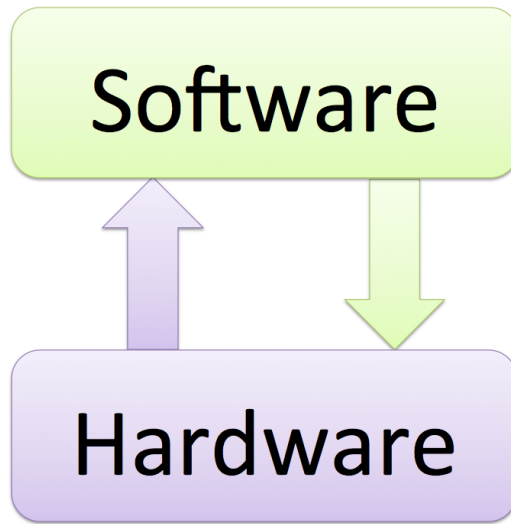


Figure 9-1: Hardware and Software working together

In Figure 2-2 we see a typical network and infrastructure that the software relies on.

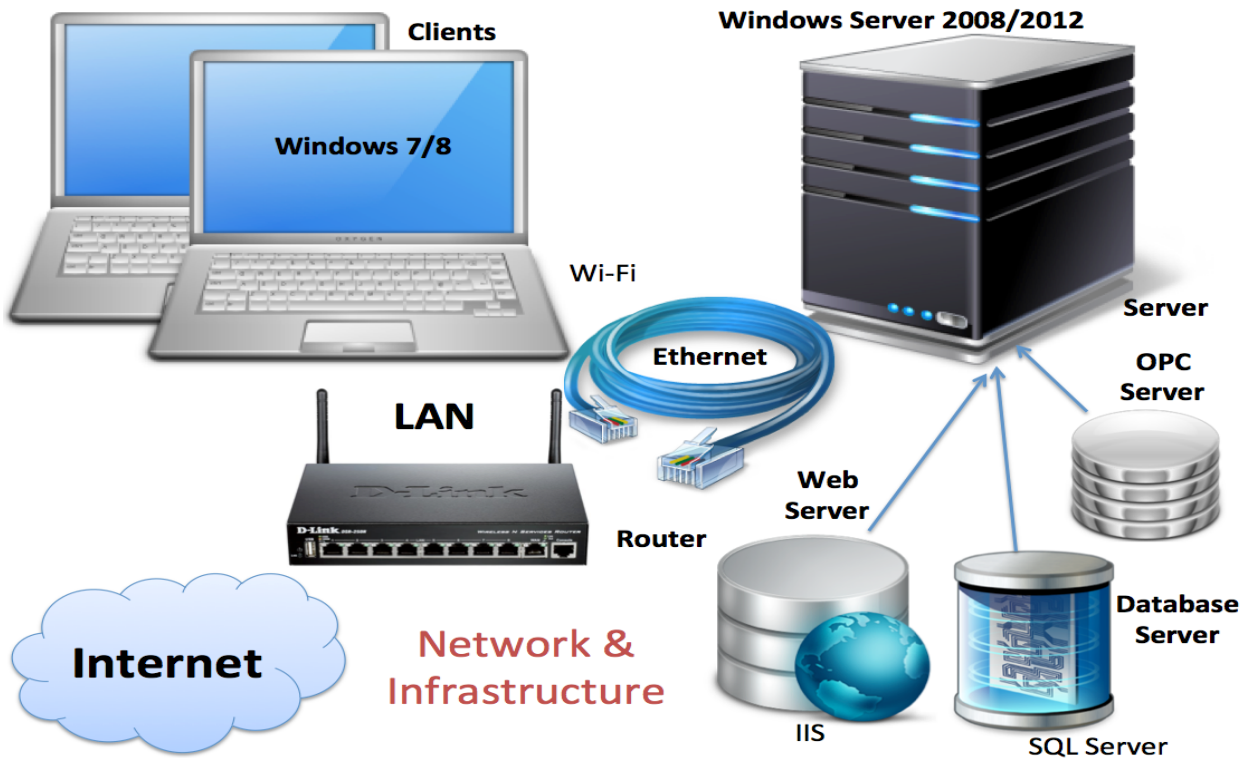


Figure 9-2: Typical Network and Infrastructure

9.1 Network

Knowledge of network topology, network components and network protocols are important within Industrial IT and Automation.

See the next chapters for details.

10 Database Systems

Web: <https://www.halvorsen.blog/documents/technology/database/>

Almost any kind of software program uses a database for back-end storage, e.g., Facebook, etc.

Not too long ago, this (Figure 10-1) was the only data-storage device most companies needed. Those days are over.



Figure 10-1: Database System in in the old days

A Database is a structured way to store lots of information. The information is stored in different tables - “Everything” today is stored in databases!

Examples:

- Bank/Account systems
- Information in Web pages such as Facebook, Wikipedia, YouTube, etc.
- Educational Systems, like Fronter, Canvas, etc.
- ... lots of other examples!

Popular Database Systems:

- Microsoft SQL Server
- Oracle
- MySQL

- SQLite
- MS Access
- MariaDB

The main focus in this chapter will be Microsoft SQL Server. For more information about database systems.

10.1 Structured Query Language (SQL)

Here we will only give a short introduction to Structured Query Language (SQL). For more information about SQL, please see (Halvorsen, 2017. Structured Query Language).

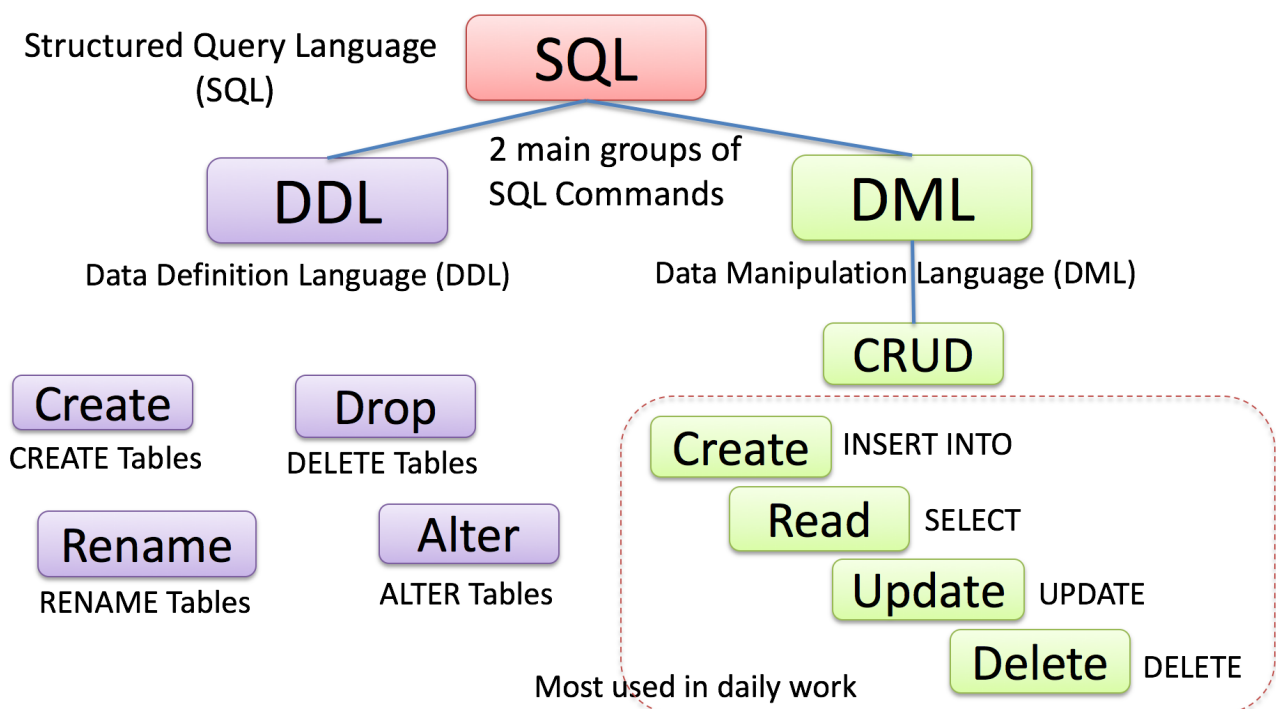


Figure 10-2: SQL Overview

SQL is a Database Computer Language designed for Managing Data in Relational Database Management Systems (RDBMS). In SQL, we have 4 different types of queries:

- INSERT
- SELECT
- UPDATE
- DELETE

Below we see some examples of typical SQL queries:

```
insert into STUDENT (Name, Number, SchoolId)
values ('John Smith', '100005', 1)
```

```
select SchoolId, Name from SCHOOL  
  
select * from SCHOOL where SchoolId > 100  
  
update STUDENT set Name='John Wayne' where StudentId=2  
  
delete from STUDENT where SchoolId=3
```

These are referred to as **CRUD** – Create (Insert), Read (Select), Update and Delete.

Here are some “Best practice” recommendations for creating tables in a database system:

- **Tables:** Use upper case and singular form in table names – not plural, e.g., “STUDENT” (not students)
- **Columns:** Use Pascal notation, e.g., “StudentId”
- **Primary Keys:**
 - If the table name is “COURSE”, name the Primary Key column “CourseId”, etc.
 - “Always” use Integer and Identity(1,1) for Primary Keys
- Specify Required Columns (NOT NULL) – i.e., which columns that need to have data or not
- **Data Types:** Standardize on these Data Types: int, float, varchar(x), datetime, bit
- Use English for table and column names
- Avoid abbreviations! (Use RoomNumber – not RoomNo, RoomNr, ...)

Advanced SQL Features:

- **Views:** Views are virtual tables for easier access to data stored in multiple tables.
- **Stored Procedures:** A Stored Procedure is a precompiled collection of SQL statements. In a stored procedure, you can use if sentence, declare variables, etc.
- **Triggers:** A database trigger is code that is automatically executed in response to certain events on a particular table in a database.
- **Functions:** With SQL and SQL Server you can use lots of built-in functions or you may create your own functions

In Figure 10-3 we see how these things are part of the Data tier in the Application.

Data Tier

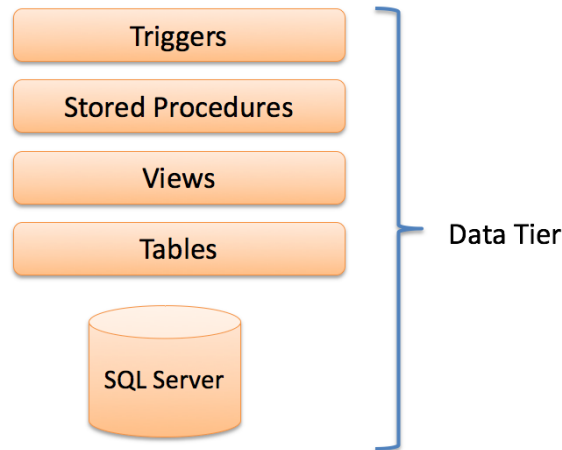
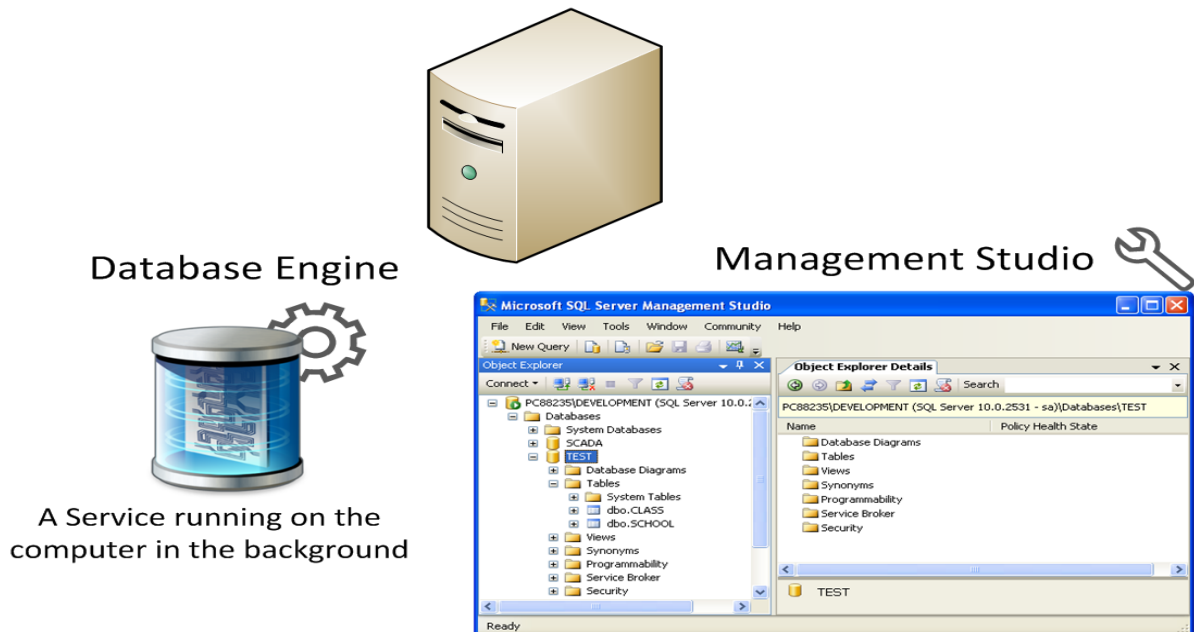


Figure 10-3: Data Tier

10.2 SQL Server

SQL Server (Figure 10-4) consists of a Database Engine and a Management Studio. The Database Engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server). The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).



A Graphical User Interface to the database used for configuration and management of the database

Figure 10-4: SQL Server

In Figure 10-5 we see the SQL Server Management Studio.

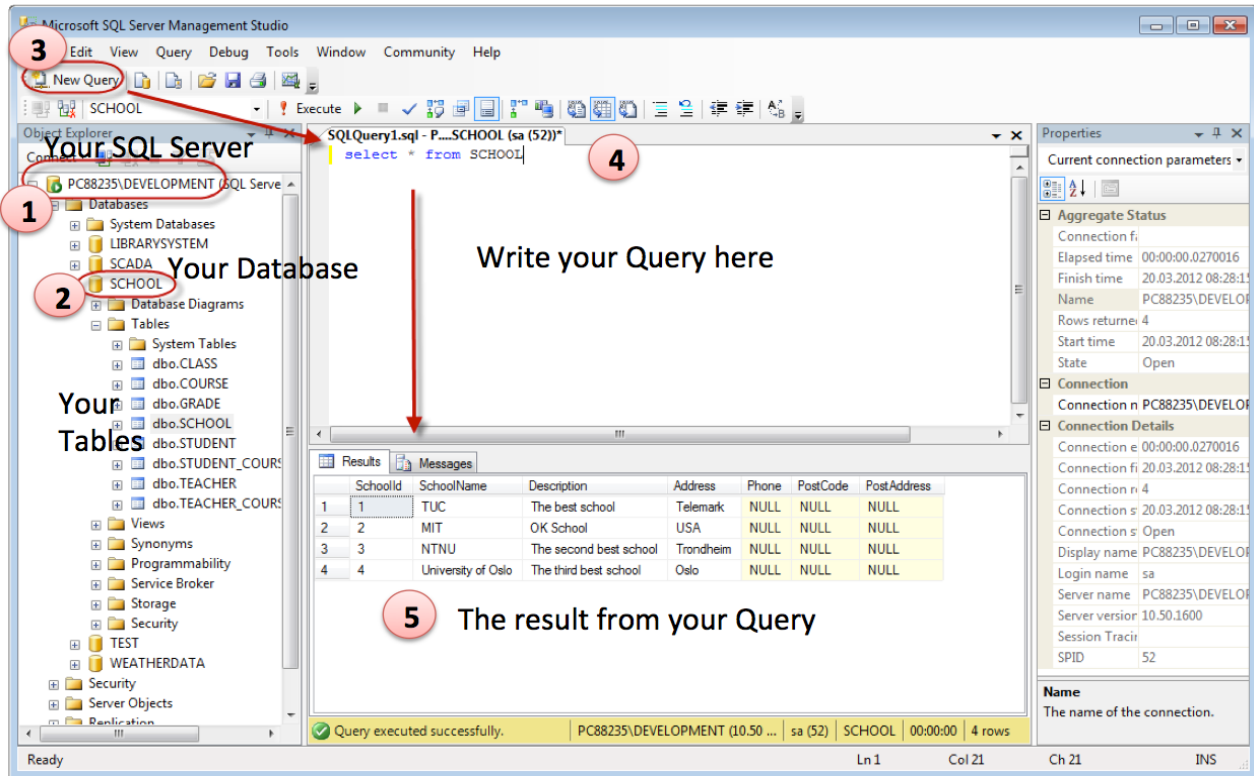


Figure 10-5: SQL Server Management Studio

10.3 ODBC

ODBC (Open Database Connectivity) is a standardized interface (API) for accessing the database from a client. You can use this standard to communicate with databases from different vendors, such as Oracle, SQL Server, etc. The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems.

Figure 10-6 shows the ODBC Data Source Administrator Tool.

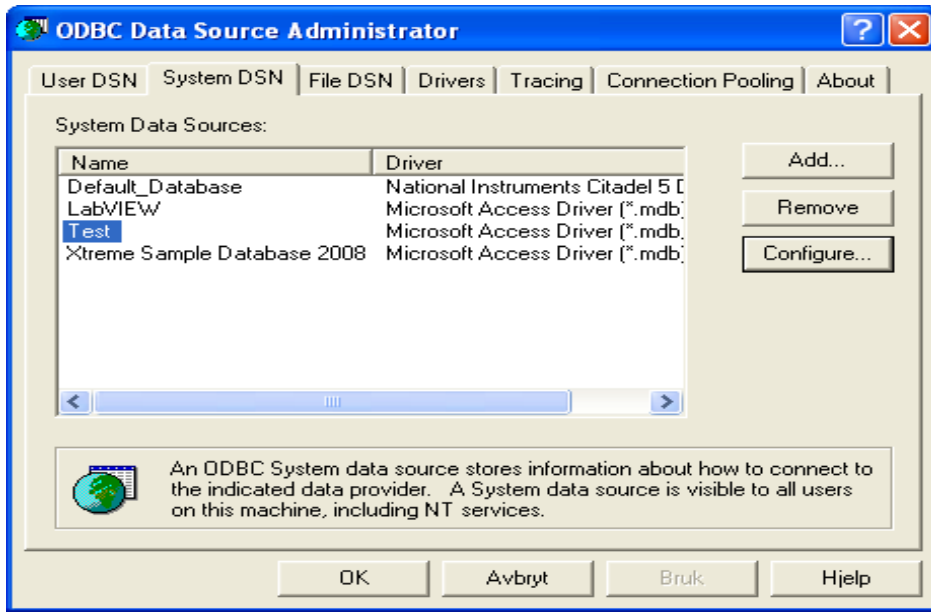


Figure 10-6: ODBC

You find it in Windows in the Control Panel: Control Panel → Administrative Tools → Data Sources (ODBC).

ODBC – Step by Step Instructions

The Name of your ODBC Connection

The Name of your SQL Server

Select the Database you are using

Use either Windows or SQL Server authentication (Windows is simplest to use!)

Test your connection to see if its works

Figure 10-7: ODBC Configuration – Step by Step

10.4 Database Communication in LabVIEW

10.4.1 LabVIEW SQL Toolkit

For Easy Database Communication with LabVIEW you can use my SQL Toolkit (Figure 10-8).

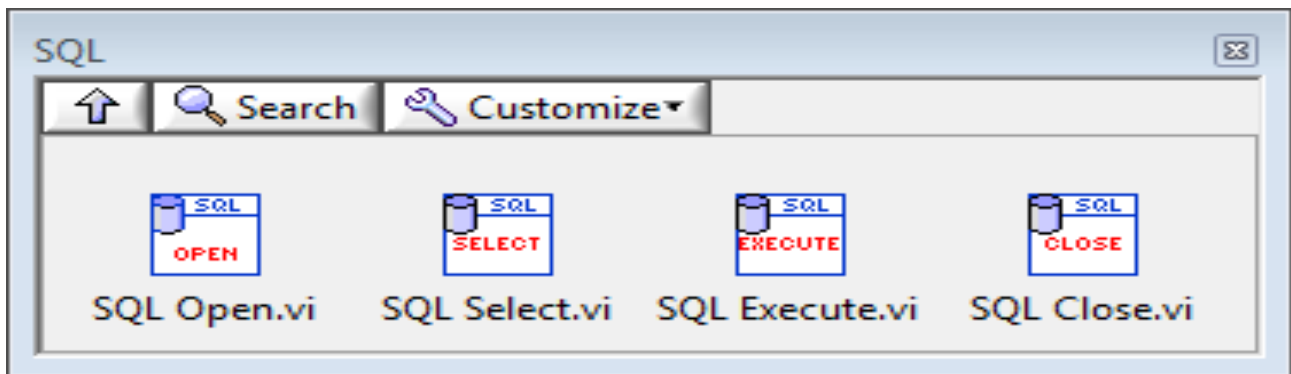


Figure 10-8: LabVIEW SQL Toolkit

Download for free here:

<https://www.halvorsen.blog/documents/technology/database/>

In Figure 10-9 we see an example where we get data from a database into LabVIEW.

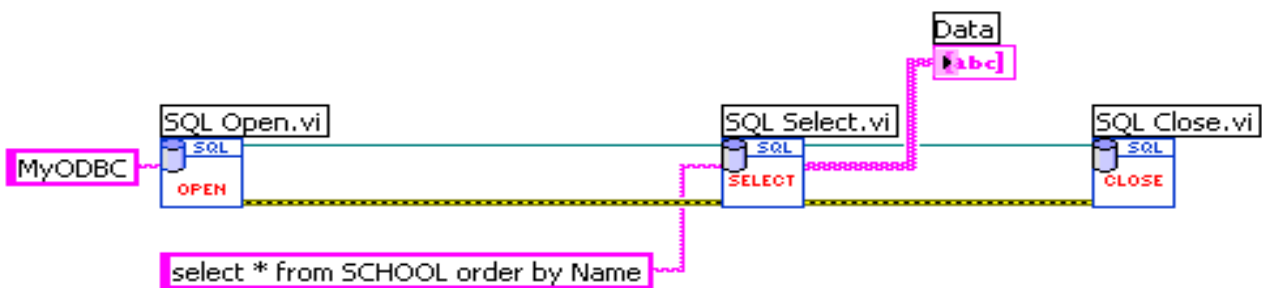


Figure 10-9: Get Data from the Database into LabVIEW

Figure 10-10 we see an example where we write data from LabVIEW into the database.

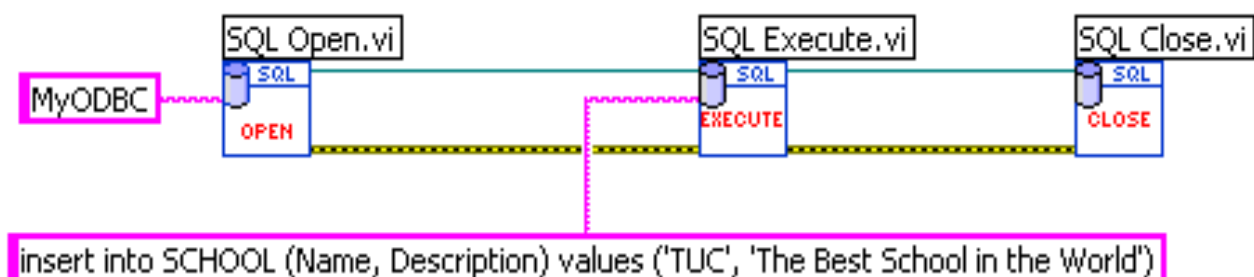


Figure 10-10: Write Data from LabVIEW into the Database

10.4.2 LabVIEW Example

In this example (Figure 10-11) we are logging temperature data from a sensor into the database.

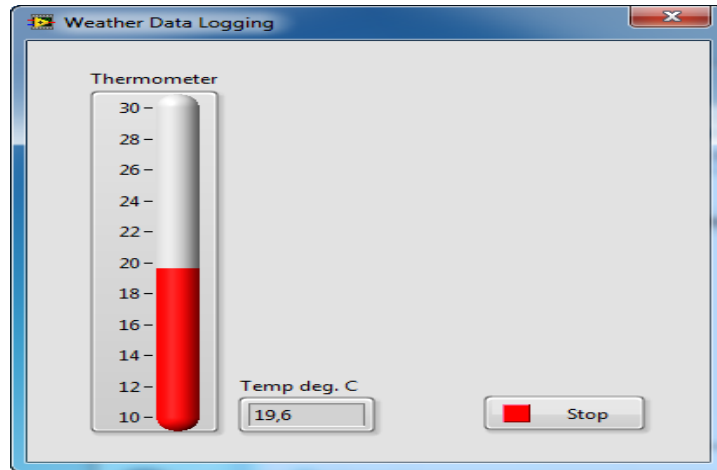


Figure 10-11: LabVIEW Example (Front Panel) – Logging Temperature Data to a Database

Figure 10-12 shows the block diagram (code).

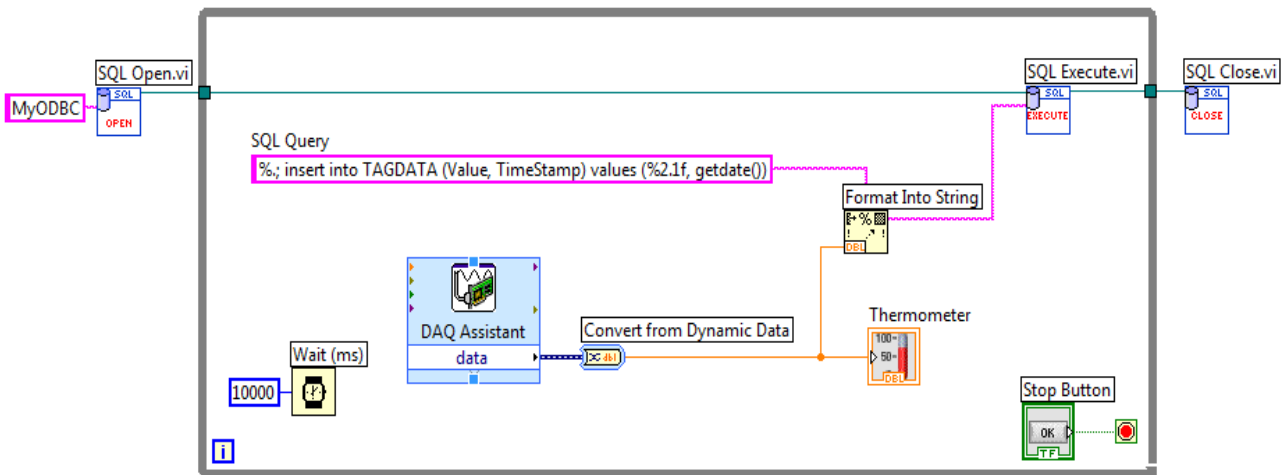


Figure 10-12: LabVIEW Example (Block Diagram) – Logging Temperature Data to a Database

10.5 Database Communication in C#

10.5.1 C# Example

Below we see an example (Code Snippet 10-1):

Code Snippet 10-1: Database Communication in C#

```
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data;

public class Book
{
    public int BookId { get; set; }
}
```

```
public string Title { get; set; }
public string Isbn { get; set; }
public string PublisherName { get; set; }
public string AuthorName { get; set; }
public string CategoryName { get; set; }
public List<Book> GetBooks(string connectionString)
{
    List<Book> bookList = new List<Book>();
    SqlConnection con = new SqlConnection(connectionString);
    string selectSQL = "select BookId, Title, Isbn, PublisherName,
AuthorName, CategoryName from GetBookData";
    con.Open();
    SqlCommand cmd = new SqlCommand(selectSQL, con);
    SqlDataReader dr = cmd.ExecuteReader();
    if (dr != null)
    {
        while (dr.Read())
        {
            Book book = new Book();
            book.BookId = Convert.ToInt32(dr["BookId"]);
            book.Title = dr["Title"].ToString();
            book.Isbn = dr["ISBN"].ToString();
            book.PublisherName = dr["PublisherName"].ToString();
            book.AuthorName = dr["AuthorName"].ToString();
            book.CategoryName = dr["CategoryName"].ToString();
            bookList.Add(book);
        }
    }
    return bookList;
}
```

For more information, details and examples, please see the Tutorial "Introduction to Visual Studio and C#" available on my Blog.

11 Web Services

A growing trend is to use a technology built on TCP and HTTP called Web Services.

A Web Service is an application programming interface (API) that can be accessed via HTTP requests. When called, Web Services return a human-readable response.

Modern Web Services use JSON responses but other response options are XML, HTML, or plain text.

What is Web Services and why do we need them?

We start with the problem (or the challenge): How do we share data between devices in a typical modern network (see Figure 11-1)?

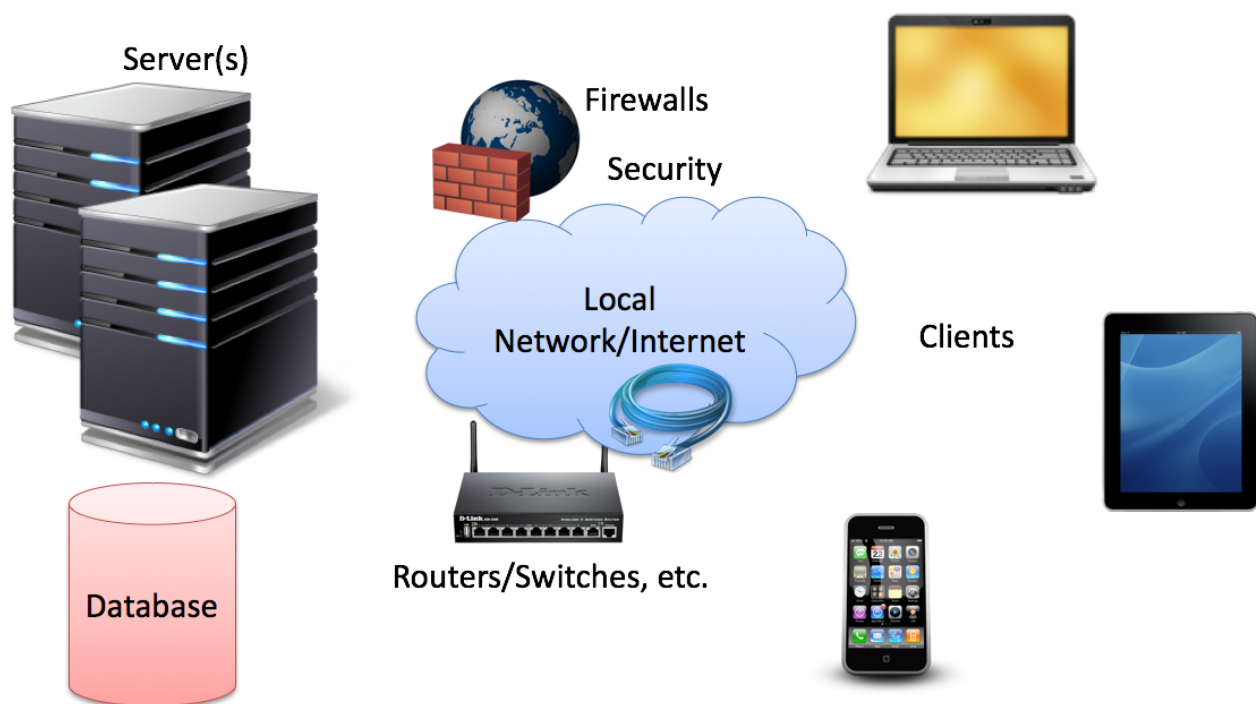


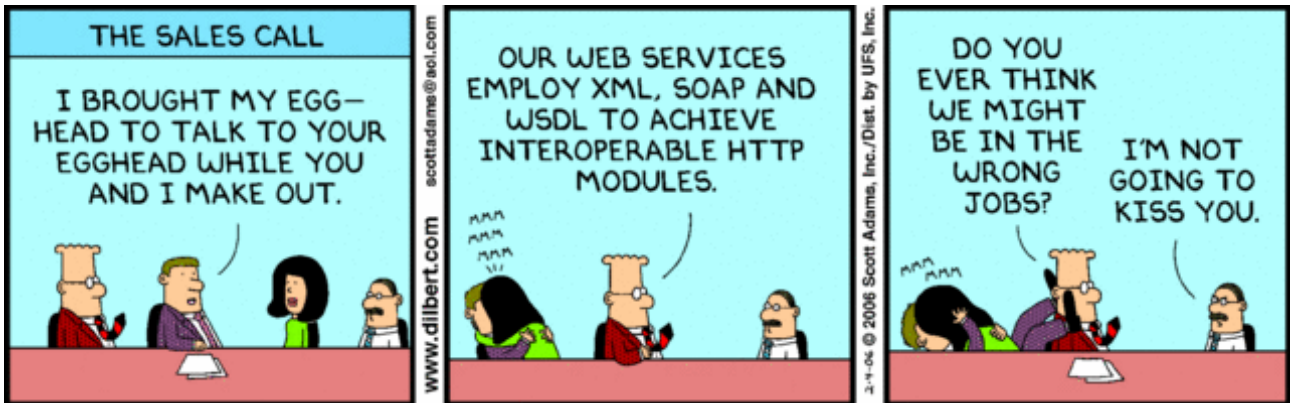
Figure 11-1: Typical Network

Direct connection between the database (which is located on a server either in a local network in on the Internet) and the clients that need the data is normally not possible, due to security, compatibility issues, etc. (Firewalls, Hacker Attacks, etc.), see Figure 11-2.



Figure 11-2: Direct Access to Database is normally not allowed

Direct connection in a Local Network (behind the Firewall) is normally OK – but not over the Internet.



So what is the solution to this problem or challenge? You have probably guessed it already, the solution is Web Services.

Web Services uses standard web protocols like HTTP, etc. HTTP is supported by all Web Browser, Servers and many Programming Languages. This means you can create and use Web Services in all the popular programming languages. See Figure 11-3.

With Web Services you have no problems with Firewalls. If you have access to the Web, you will have access to your Data.

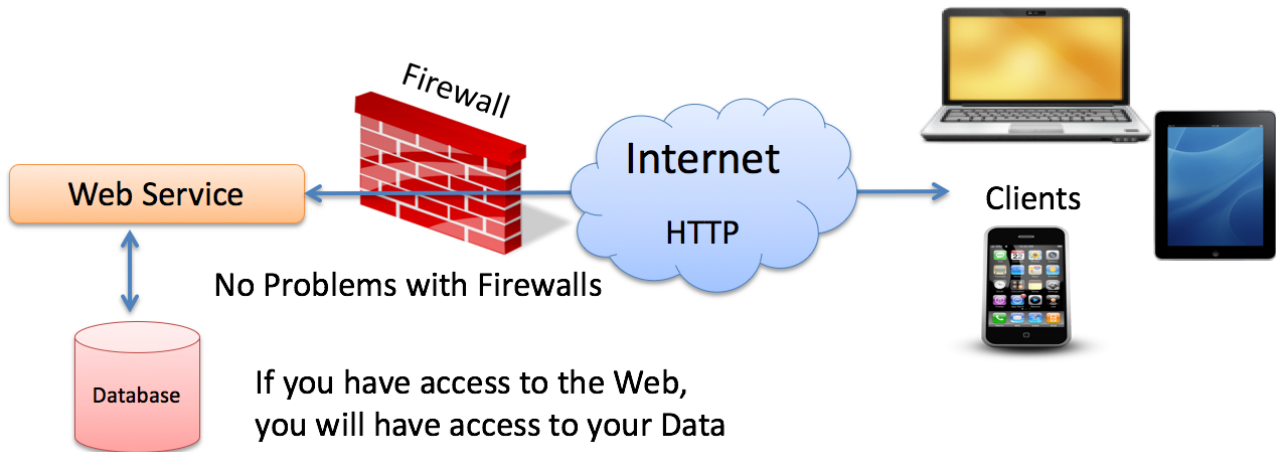


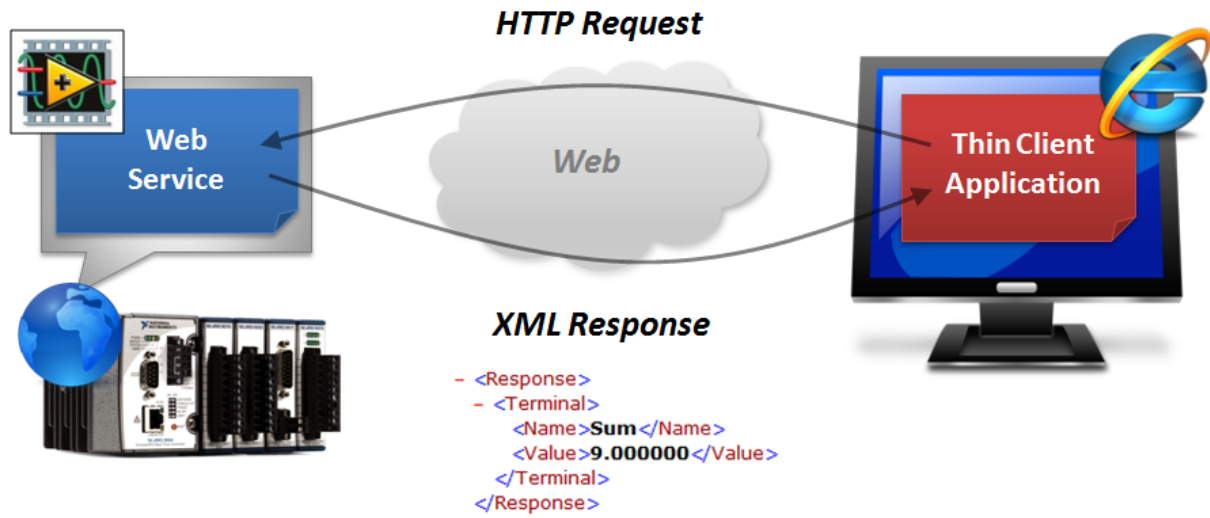
Figure 11-3: Web Service

There are several reasons why engineers and IT departments are choosing Web Services over other communication technologies.

- First, Web Services are straightforward and simple to access from any programming language, including C++, Objective C, C# and LabVIEW.
- Second, since Web Services sit on top of common network protocols, the communication is considered “IT friendly” compared with proprietary network protocols.
- Third, Web Services can also be easily encrypted via industry-standard technologies like Secure Sockets Layer (SSL) or Transport Layer Security (TLS).

11.1 Web Services with LabVIEW

LabVIEW has full support for both creating Web Services as well as consuming Web Services. See Figure 11-4.

Device with I/O (Server)**Client Machine****Figure 11-4: LabVIEW Web Service**

In LabVIEW, you create a Web Service from the Project Explorer. You need to create either a New LabVIEW Project or use your existing Project.

In the Project Explorer, you need to right-click on “My Computer” and then select New->Web Service (see Figure 11-5).

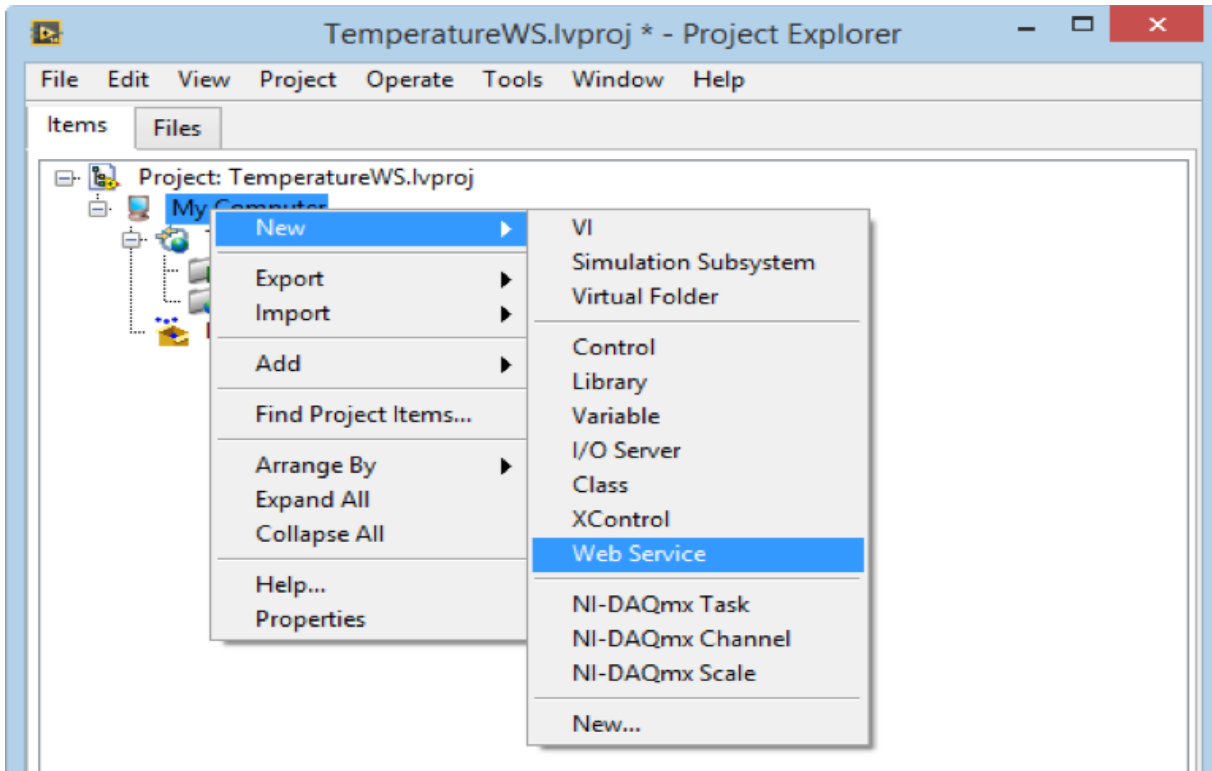


Figure 11-5: Creating Web Service in LabVIEW

Next, you need to create one or more Web Service Methods. See Figure 11-6.

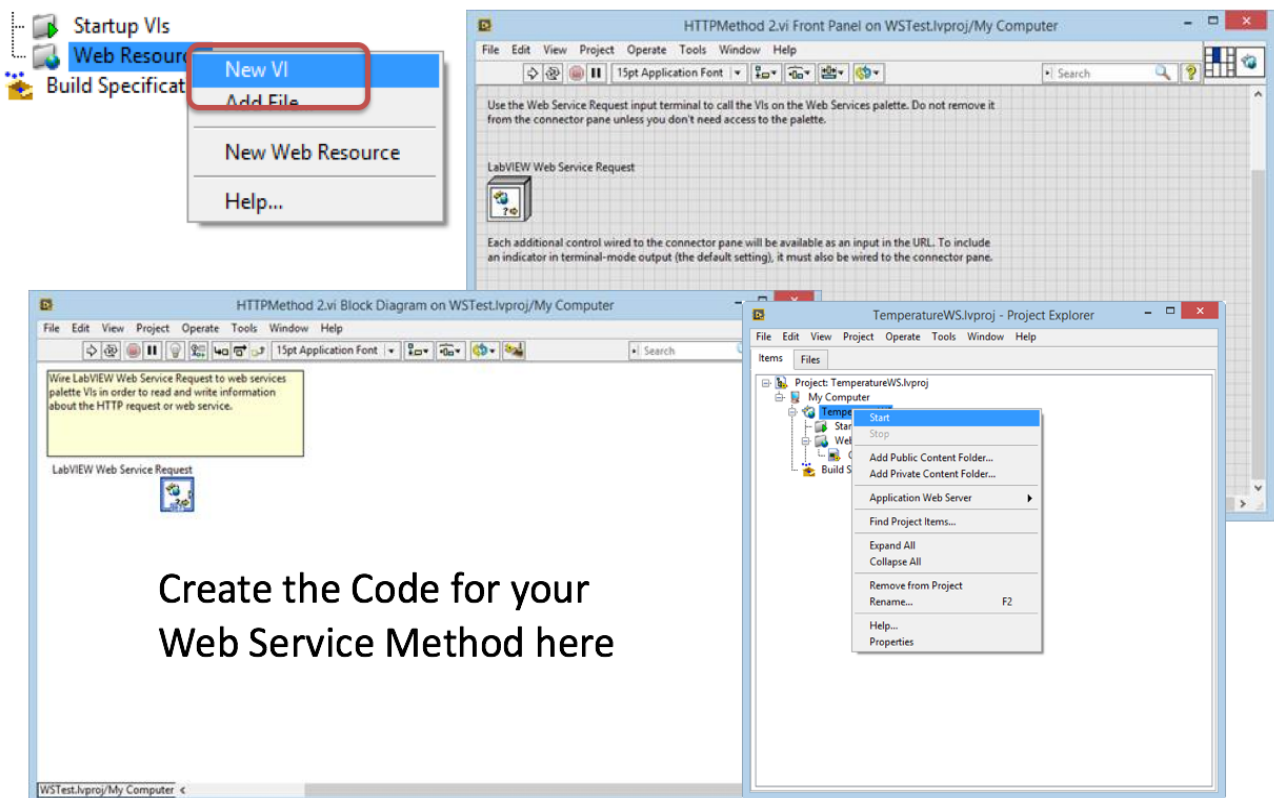


Figure 11-6: Create a Web Service Method

Then you need to create the code (your Block Diagram) for your Web Service Method.

In this example, we just make a simple temperature simulator that calculates a random temperature value between 20 and 50 degrees Celsius. See Figure 11-7.

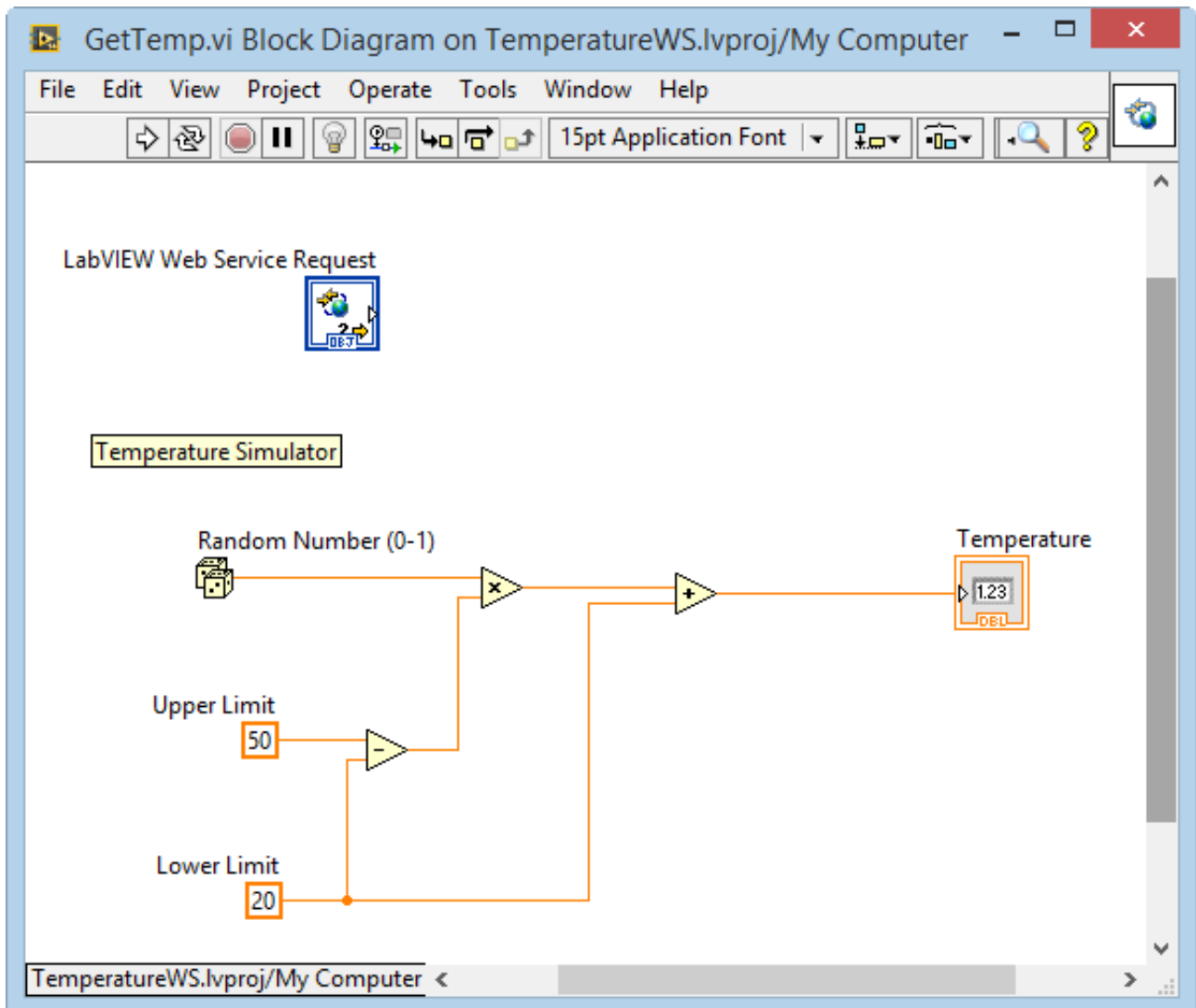


Figure 11-7: LabVIEW Web Service Method Example

Note! There is NOT need for a While Loop (neither a Stop/Exit button) in your LabVIEW Web Service Method, because these things are handled by the Web Service itself.

When you have created your code/block diagram, you need to define inputs and outputs, or the interface. You do this in the same way as you define inputs and outputs in a LabVIEW SubVI.

Note! There is no need for a beautiful GUI since the user will not be able to see the Front Panel.

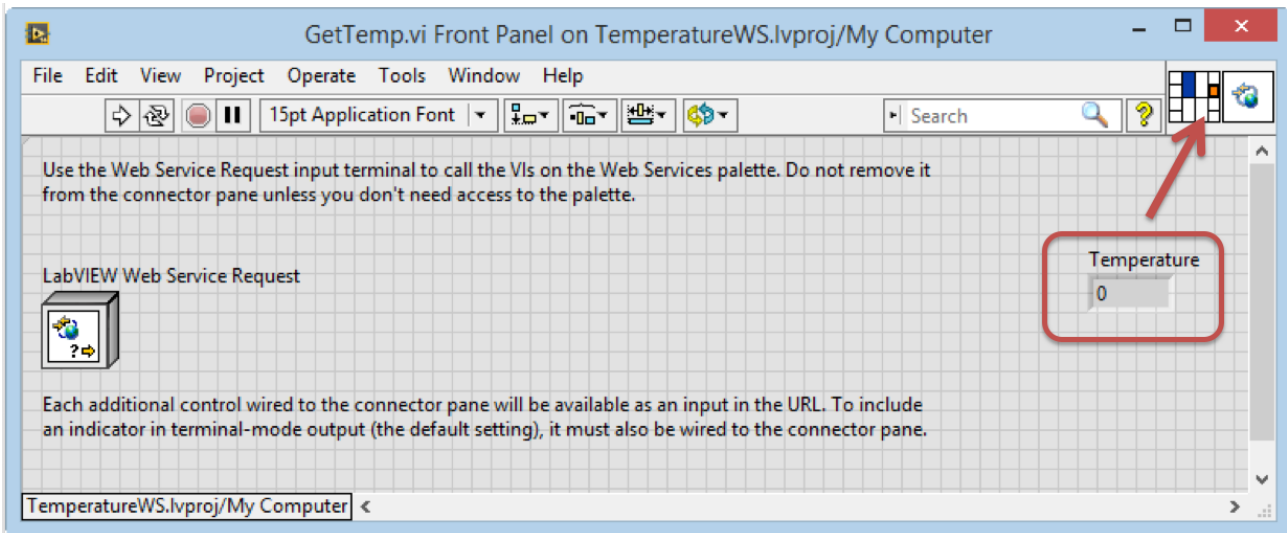


Figure 11-8: LabVIEW Web Service Method Front Panel

You are now ready to test the LabVIEW Web Service. Right-click and select “Start” in order to start the Web Service in “Debug/Test” mode.

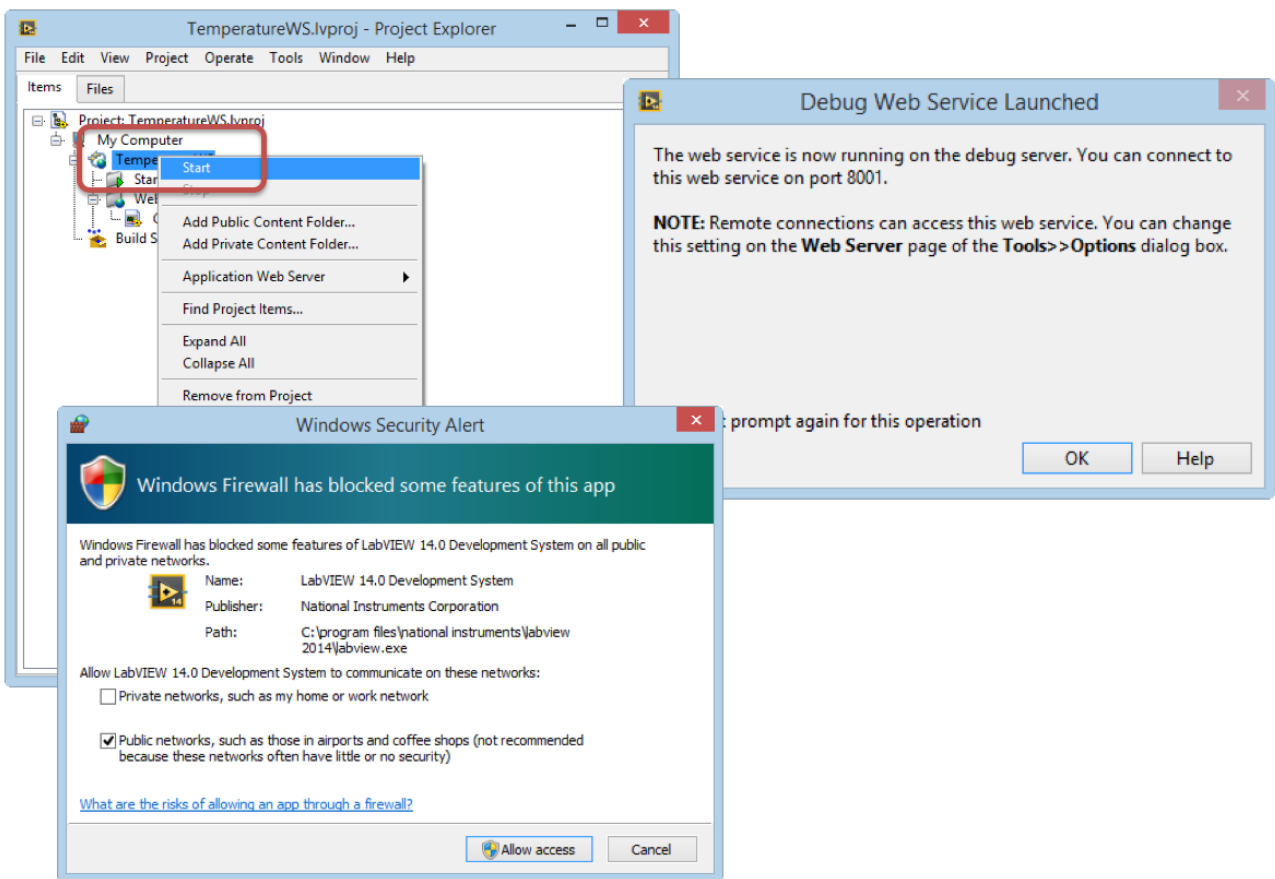


Figure 11-9: Testing the Web Service

Make sure to allow access in the Firewall.

Now you are ready to see if the Web Service is working as expected. If you don't know the URL for the Web Service, you can right-click and select "Show Method URL...".

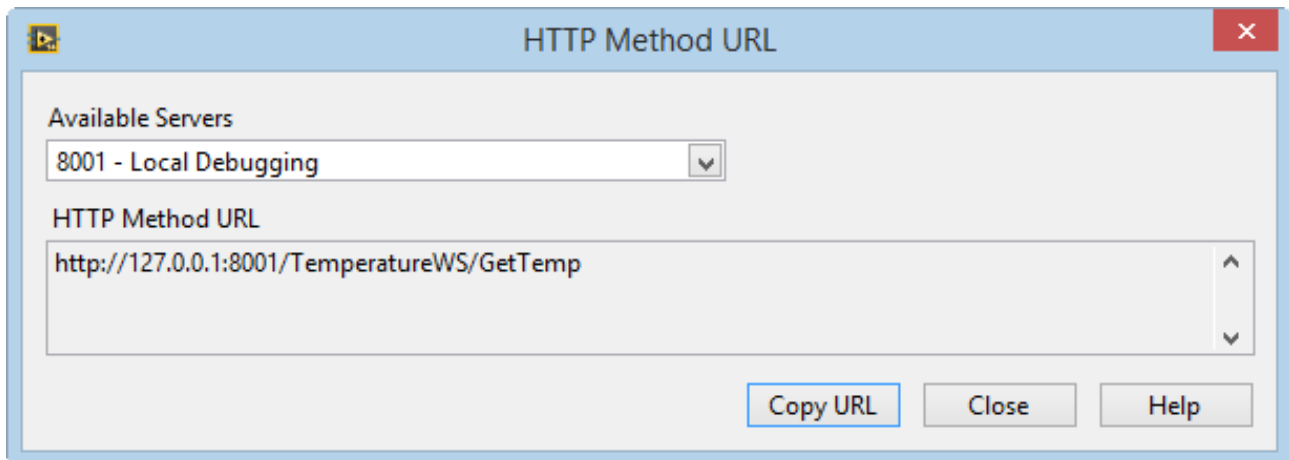


Figure 11-10: Show Method URL Dialog Box

Enter (or past) the URL in your Web Browser to see if it works.

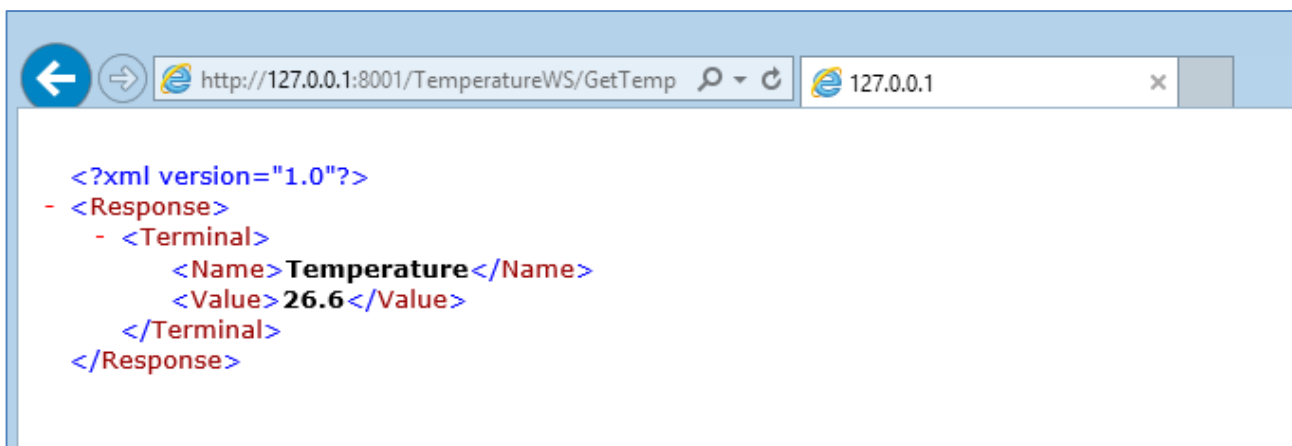


Figure 11-11: Testing the Web Service from your Web Browser

If you hit "Refresh", you should see the temperature value is changing.

Final Step: Publish the Web Service.

When you are finished, you need to Publish the Web Service to the Web Server

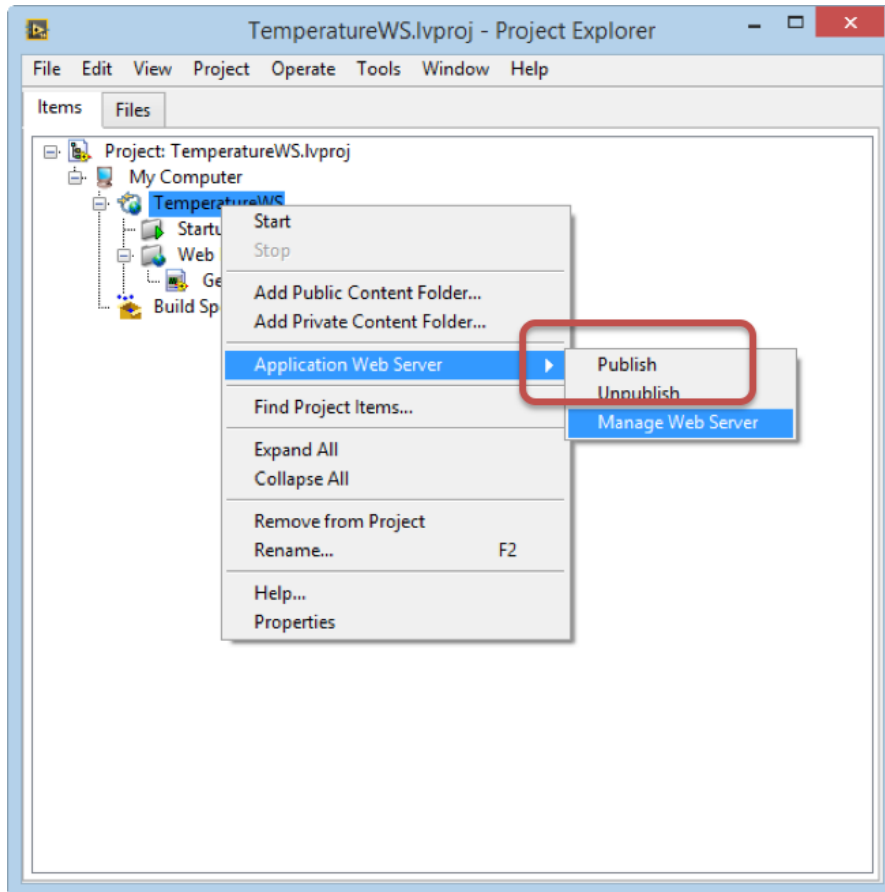


Figure 11-12: Publish the Web Service to the Web Server

You find more resources about Web Services and LabVIEW Web Services on the Web page.

11.2 Data Dashboard for LabVIEW

Data Dashboard for LabVIEW (see Figure 11-13) is an App for iOS/Android where you can create custom Dashboards (HMI) to Control and Monitor your LabVIEW application remotely using LabVIEW Web Services. You get the best experience on the iPad or Android Tablets.

Data Dashboard for LabVIEW

[View More by This Developer](#)

By National Instruments

Open iTunes to buy and download apps.



[View in iTunes](#)

Free

Category: [Productivity](#)
Updated: Sep 29, 2014
Version: 2.2.1
Size: 48.2 MB
Language: English
Seller: National Instruments
© 2014 National Instruments Corporation
[Rated 4+](#)

Compatibility: Requires iOS 7.0 or later. Compatible with iPad.

Description

The Data Dashboard for LabVIEW app lets you create a custom dashboard that can remotely control and monitor running NI LabVIEW applications. This is done by networking deployed NI Shared Variables, LabVIEW Web Services, or the NI Technical Data Cloud with visual objects like graphs, meters, and switches among many other available

[National Instruments Web Site](#) > [Data Dashboard for LabVIEW Support](#) > [Application License Agreement](#) > [...More](#)

What's New in Version 2.2.1

- Updated look for iOS 7 and 8
- iOS 8 compatibility
- Various bug fixes

iPad Screenshots

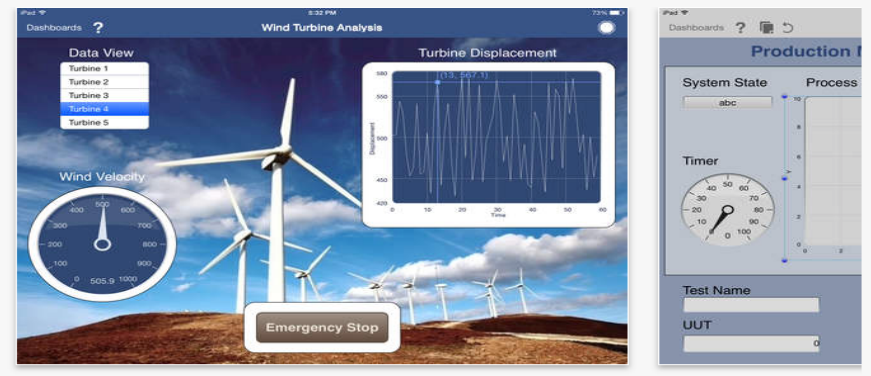


Figure 11-13: Data Dashboard for LabVIEW available on Smartphones and Tablets

Weather Station Example:

Figure 11-14 we see an example where Data Dashboard for LabVIEW has been used for presenting weather data from the Weather Station we have at the university.

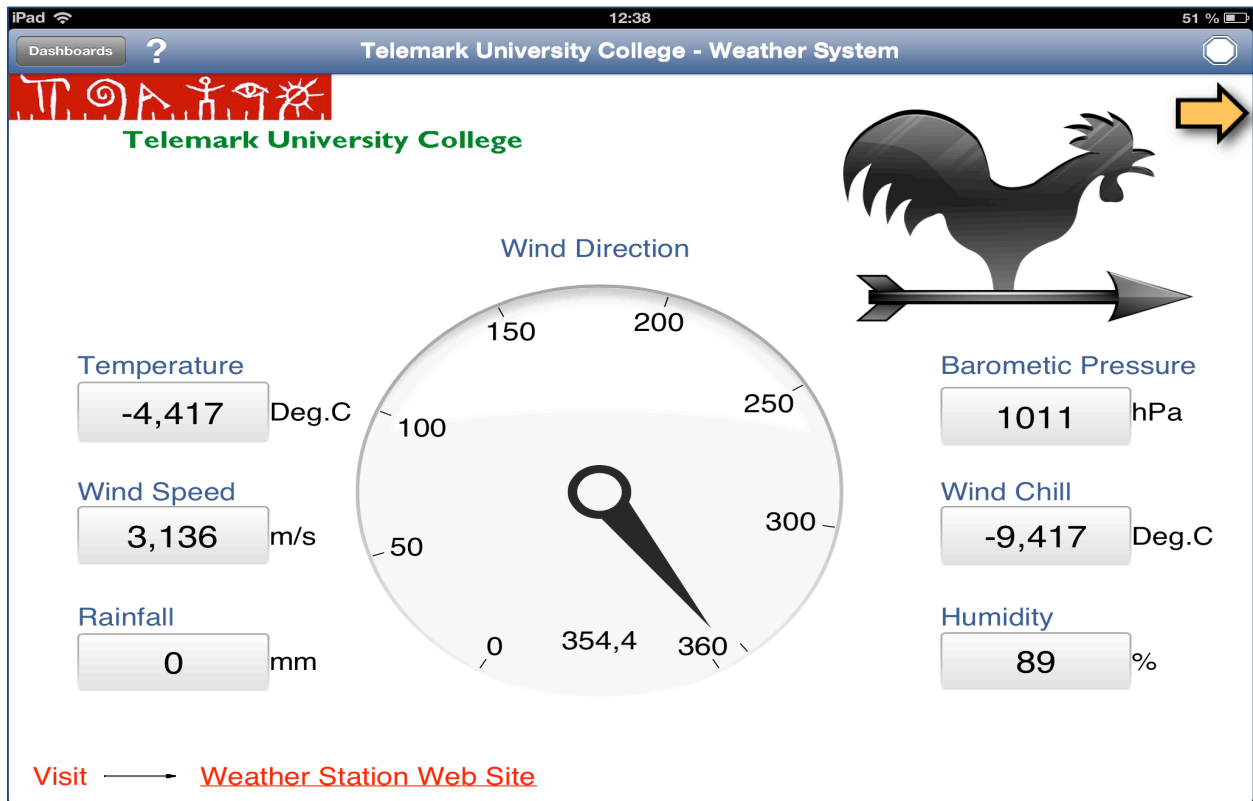


Figure 11-14: Weather Station Data Dashboard

12 Modbus

12.1 What is Modbus?

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs).

Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices

The development and update of Modbus protocols has been managed by the Modbus Organization since April 2004, when Schneider Electric transferred rights to that organization

Modbus became the first widely accepted fieldbus standard.

Modbus protocol is defined as a master/slave protocol, meaning a device operating as a master will poll one or more devices operating as a slave.

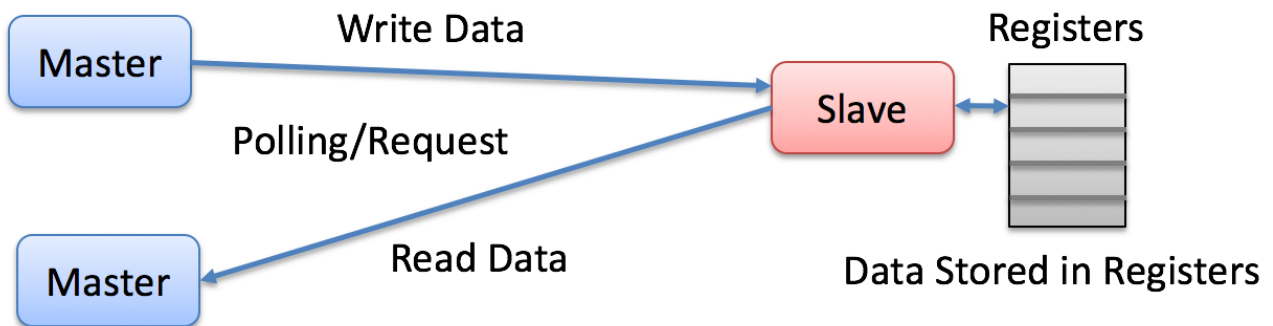


Figure 12-1: Modbus is a Master/Slave Protocol

This means a slave device cannot volunteer information; it must wait to be asked for it.

The master will write data to a slave device's registers, and read data from a slave device's registers. A register address or register reference is always in the context of the slave's registers.

References:

- Modbus Organization: <http://www.modbus.org>
- Modbus (Wikipedia): <https://en.wikipedia.org/wiki/Modbus>
- Introduction to Modbus (National Instruments): <http://www.ni.com/white-paper/7675/en/>

- Modbus 101 - Introduction to Modbus:
http://www.csimn.com/CSI_pages/Modbus101.html
- Modbus TCP/IP: <http://www.rtaautomation.com/technologies/modbus-tcpip/>
- Modbus RTU: <http://www.rtaautomation.com/technologies/modbus-rtu/>

The master typically is a PLC (Programmable Logic Controller), PC or DCS (Distributed Control System). A remote terminal unit (RTU) is a microprocessor-controlled electronic device that interfaces objects in the physical world to a DCS or SCADA System. Modbus RTU slaves are often field devices, all of which connect to the network in a multi-drop configuration. See Figure 12-2.

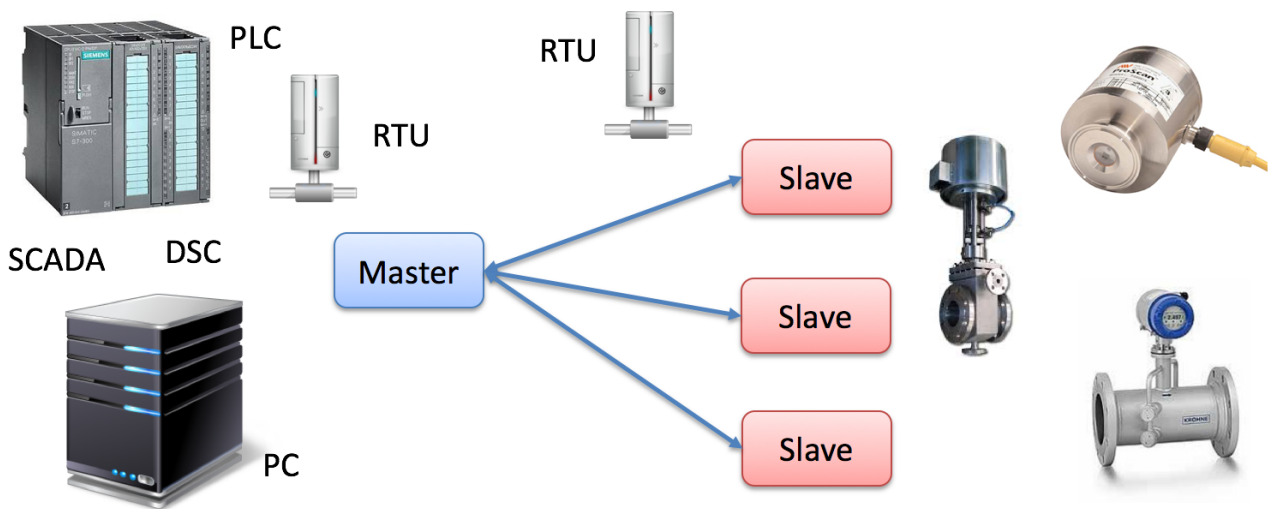


Figure 12-2: Modbus Overview

12.2 Modbus Register Types

Modbus supports 4 different types of registers:

- Coil (Discrete Output)
 - Coils are 1-bit registers, used to control discrete outputs, Read or Write
- Discrete Input
 - 1-bit registers
- Input Register
 - 16-bit data registers
- Holding Register
 - 16-bit data registers

Register Addresses:

0x = **Coil**, Address Range: **00001-09999**

1x = **Discrete Input**, Address Range: **10001-19999**

3x = **Input Register**, Address Range: **30001-39999**

4x = **Holding Register**, Address Range: **40001-49999**

When using the extended referencing, all *number* references must be exactly six digits. This avoids confusion between coils and other entities. For example, to know the difference between holding register #40001 and coil #40001, if coil #40001 is the target, it must appear as #040001.

12.2.1 Access Levels

In SCADA systems, it is common for embedded devices to have certain values defined as inputs, such as gains or proportional integral derivative (PID) settings, while other values are outputs, like the current temperature or valve position. To meet this need, Modbus data values are divided into four ranges

In many cases, sensors and other devices generate data in types other than simply Booleans and unsigned integers. It is common for slave devices to convert these larger data types into registers. For example, a pressure sensor may split a 32-bit floating point value across two 16-bit registers.

See Figure 12-3 for the access levels on the different types in Modbus.

Memory Type	Data Type	Master Access	Slave Access
Coils	Bit (Boolean)	Read/Write	Read/Write
Discrete Input	Bit (Boolean)	Read-only	Read/Write
Input Register	Unsigned Word	Read-only	Read/Write
Holding Register	Unsigned Word	Read/Write	Read/Write

Figure 12-3: Access Levels

12.3 Modbus Protocols

There are 3 main types of Modbus Protocols:

- Modbus ASCII
- Modbus RTU (Remote Terminal Unit)

- Modbus RTU uses RS-485 or RS-232
- Modbus TCP/IP
 - Modbus TCP uses Ethernet

Modbus ASCII and Modbus RTU are simple serial protocols that use RS-232 or RS-485 to transmit data packets.

Modbus TCP/IP follows the OSI Network Model and can be used in an ordinary Ethernet network

12.3.1 Modbus ASCII

Modbus ASCII uses ASCII characters for protocol communication.

12.3.2 Modbus RTU

Modbus RTU uses binary representation of the data for protocol communication.

Modbus RTU uses RS-485 or RS-232.

Modbus RTU requires that you know or define baud rate, character format (8 bits no parity, etc.), and slave ID (aka slave address, unit number, unit ID). A mismatch in any of these will result in no communication.

12.3.3 Modbus TCP/IP

Modbus TCP/IP follows the OSI Network Model and can be used in an ordinary Ethernet network.

Modbus TCP requires that you know or define IP addresses on the network.

Modbus TCP/IP uses Port **502**

In Modbus TCP/IP we normally use the terms Server/Client instead of Slave/Master

Slave -> Server

Master -> Client

12.4 Modbus in LabVIEW

3 ways to use Modbus in LabVIEW:

- Use a high-level OPC Server

- Use Modbus I/O Server
- Use the LabVIEW Modbus API

“LabVIEW Real-Time Module” or “LabVIEW DSC Module” required

Introduction to Modbus (National Instruments): <http://www.ni.com/white-paper/7675/en/>

12.4.1 LabVIEW Modbus API

Figure 12-4 shows the LabVIEW Modbus API palette in LabVIEW.

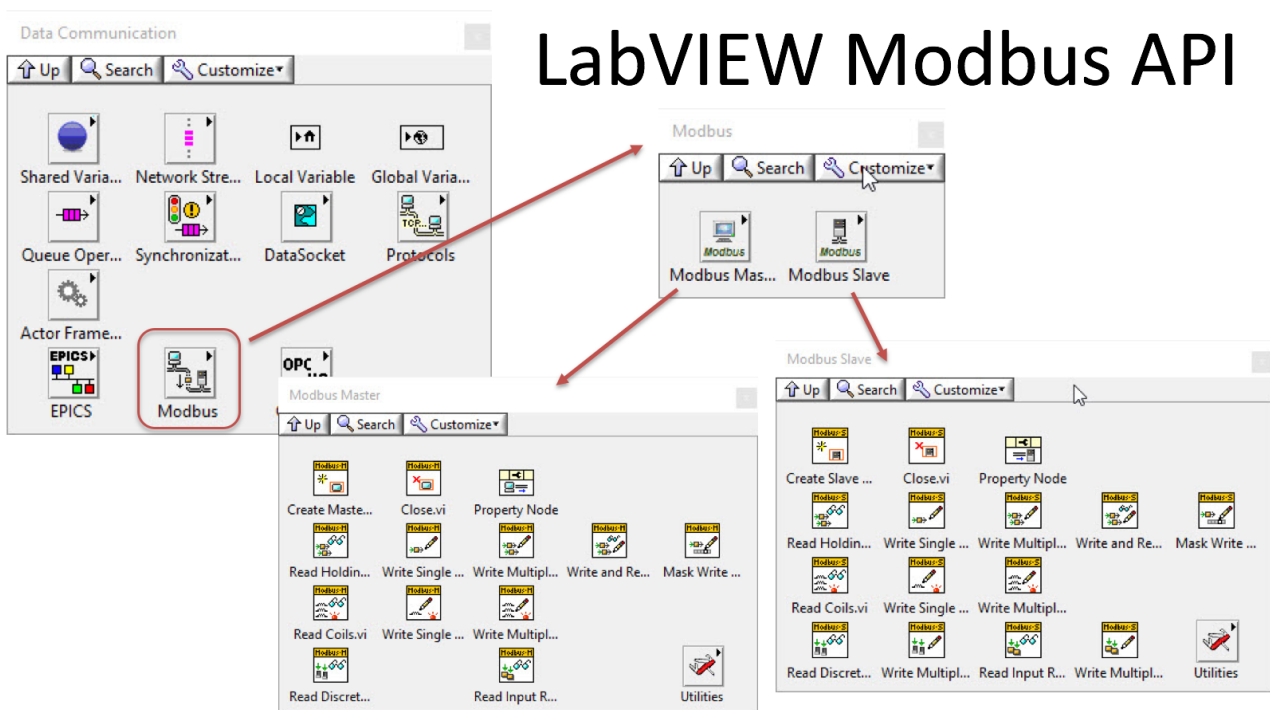


Figure 12-4: LabVIEW Modbus API

LabVIEW Example:

We will go through a simple Modbus example in LabVIEW where we create 3 different applications. In LabVIEW Application #1 we create a Modbus Slave. LabVIEW Application #2 will be a Modbus Master application where we write data to the slave, while LabVIEW Application #3 is a Modbus Master application where we read the same data as Application #2 is writing to the slave.

See Figure 12-5 for a simple sketch of the LabVIEW example.

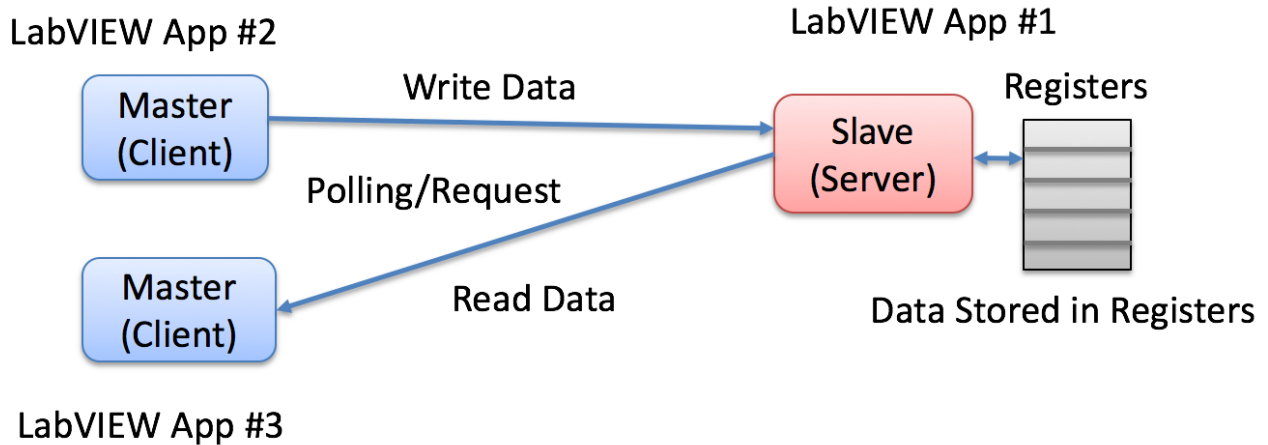
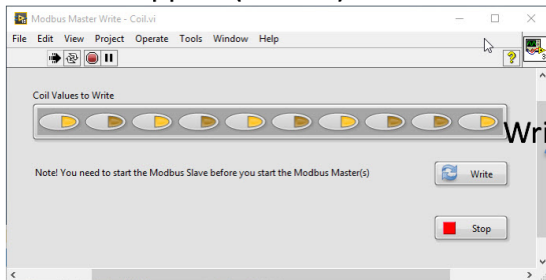


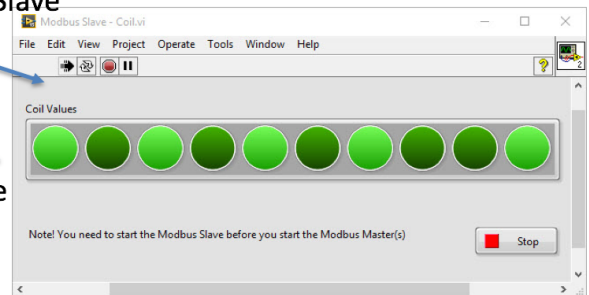
Figure 12-5: LabVIEW Modbus Example Overview

Figure 12-6 shows the Front Panel (GUI) of the 3 applications in the example.

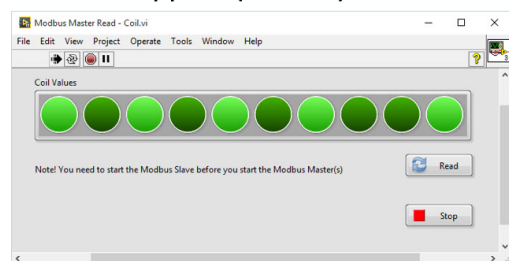
LabVIEW App #2 (Master)



LabVIEW App #1 (Slave)



LabVIEW App #3 (Master)



Write Data to Slave

Read Data from Slave

Note! You need to start/run the Modbus Slave App before you start the Modbus Master Apps

Figure 12-6: LabVIEW Example – Front Panels

In LabVIEW Application #2 we can turn on/off some Boolean switches and when pushing the “Write” button the values are pushed and store in the slave Coil Register (LabVIEW Application #1). In LabVIEW Application #1 we just continuously read the values stored in the Slave Coil Register. Dark green in Boolean True and light green is Boolean False. In LabVIEW Application #2 we see that we are able to read the values stored in the Slave Registers.

In this example, all 3 applications running on the same computer, while in practical applications these 3 applications will typically run on 3 different computers or devices.

Modbus Slave Block Diagram (Figure 12-7):

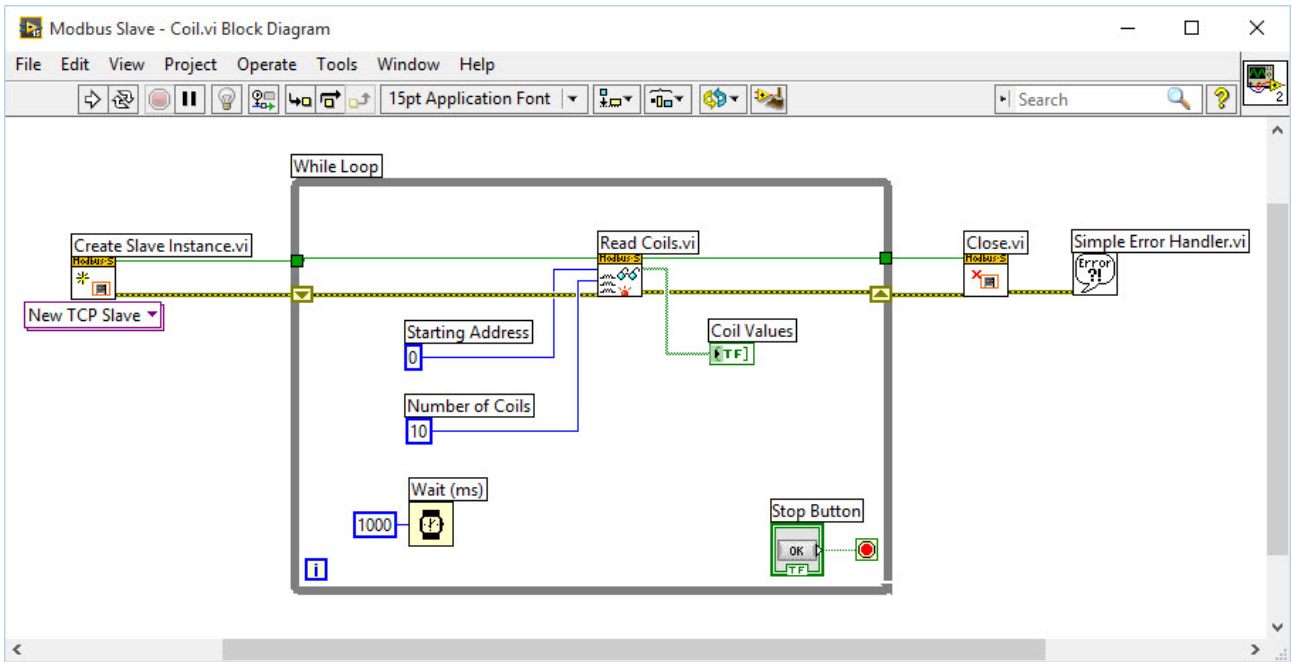


Figure 12-7: Modbus Slave Block Diagram

Modbus Master Write Block Diagram (Figure 12-8):

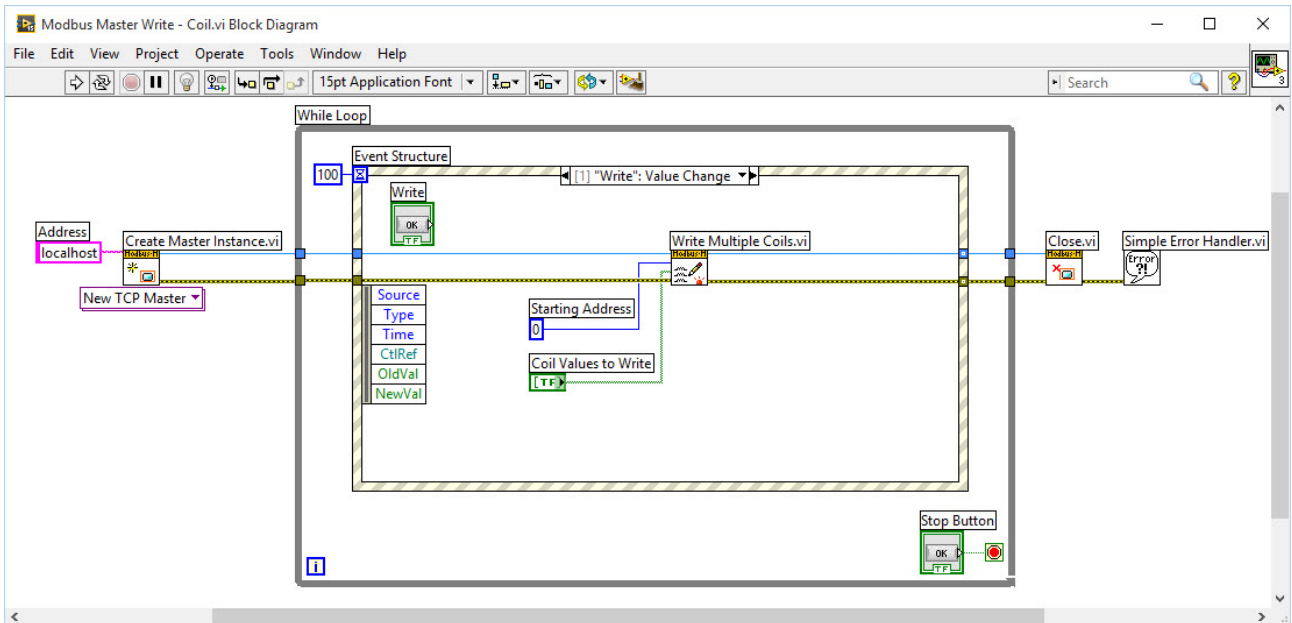


Figure 12-8: Modbus Master Write Block Diagram

Modbus Master Read Block Diagram (Figure 12-9):

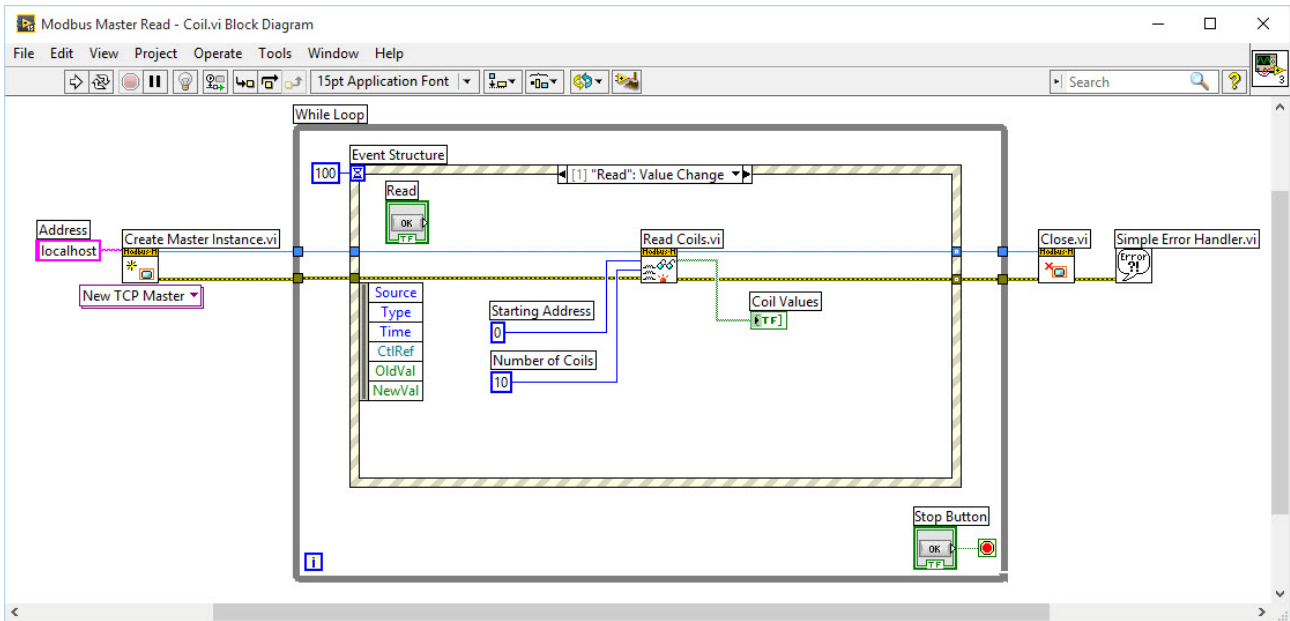


Figure 12-9: Modbus Master Read Block Diagram

12.4.2 LabVIEW Modbus Simulator

The LabVIEW Modbus Simulator is integrated with “LabVIEW Real-Time Module” or “LabVIEW DSC Module” It can be used for test purpose, etc. The LabVIEW Modbus Simulator is a Modbus Slave
In Figure 12-10 we use the LabVIEW Modbus Simulator in order to test our LabVIEW code.

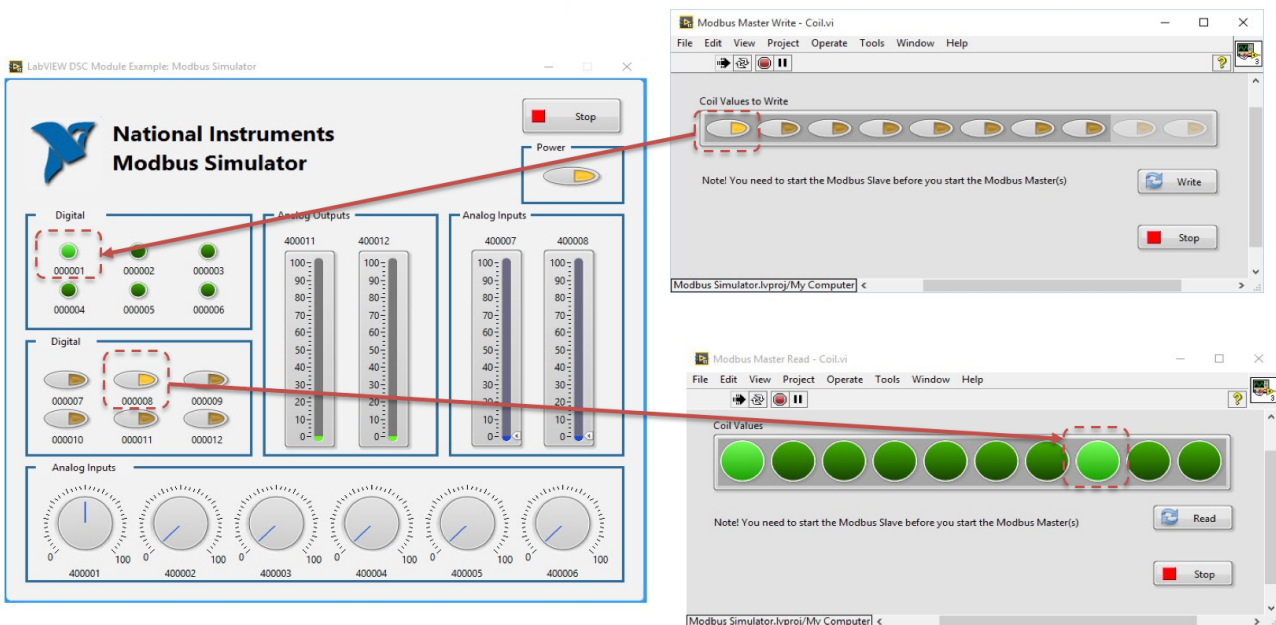


Figure 12-10: LabVIEW Modbus Simulator

You will find the LabVIEW Modbus Simulator and other Modbus applications and examples using the “NI Example Finder”.

13 Virtualization

13.1 Introduction

What is Virtualization? [Wikipedia] Virtualization, in computing, refers the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources.

The term "virtualization" traces its roots to 1960s mainframes, during which it was a method of logically dividing the mainframes' resources for different applications. Since then, the meaning of the term has evolved to the aforementioned.

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.

In hardware virtualization, the **host machine** is the actual machine on which the virtualization takes place, and the **guest machine** is the virtual machine.

The words host and guest are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine.

The software or firmware that creates a virtual machine on the host hardware is called a **hypervisor** or Virtual Machine Manager.

A **snapshot** is the state of a virtual machine, and, generally, its storage devices, at an exact point in time. Snapshots are "taken" by simply giving an order to do so at a given time, and can be "reverted" to on demand, with the effect that the VM appears (ideally) exactly as it did when the snapshot was taken.

A lot of Virtualization Software exists. Here are some examples:

- VMware Workstation/VMware Workstation Player
- VMware vSphere
- Microsoft Hyper-V
- VirtualBox
- VMware Fusion
- etc.

These (and others) are explained more in detailed later in this document.

VMware is a company that has been specializing within virtualization software.

<http://www.vmware.com>

13.2 VMware

13.2.1 VMware Workstation Player

VMware Workstation Player (Figure 13-1) is for personal use on your own PC. VMware Player is free of charge for personal noncommercial use.



Figure 13-1: VMware Workstation Player

With VMware Workstation Player, you can only run one Virtual Machine at the same time.

In Figure 13-2 we see Windows Server 2012 R2 running within the VMware Workstation Player.

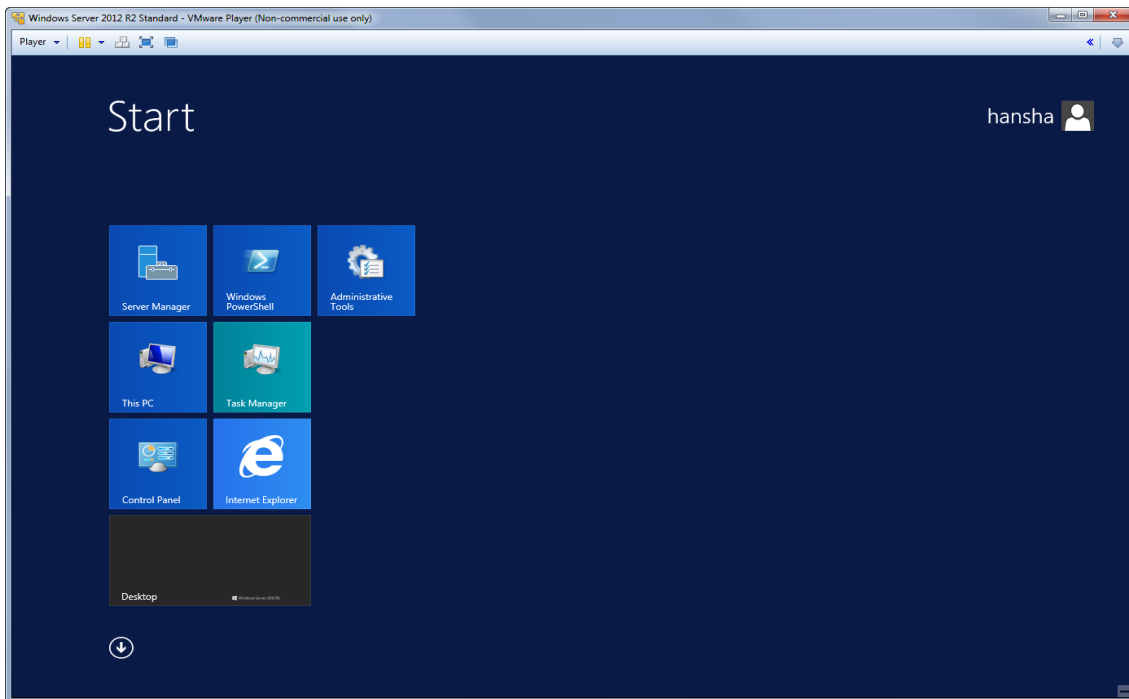


Figure 13-2: Windows Server running as a Virtual Machine

VMware Workstation Player Plus has the same functionality as VMware Player but can be used for commercial use.

Note! the free version called VMware Workstation Player was previously called VMware Player.

13.2.2 VMware Workstation

VMware Workstation is for personal (you typically install this software on a personal computer, not a server) use and if you need more features than VMware Workstation Player can offer. With VMware Workstation you can add snapshot, restore to snapshot, copy virtual machines, etc.

With VMware Workstation, you can run multiple Virtual Machines at the same time.

Test (in Norwegian): <http://www.idg.no/pcworld/article277809.ece>

13.2.3 VMware vSphere

VMware vSphere is for companies that uses virtualization in a large scale.

13.3 Microsoft Hyper-V

Hyper-V is the virtualization solution from Microsoft.

We have 3 different alternatives:

- Windows Server with Hyper-V
- Hyper-V Server
- Windows Client Hyper-V

13.3.1 Windows Server with Hyper-V

Here you need “Windows Server 2013 R2 Standard” or “Windows Server 2012 R2 Datacenter” as the host operating system.

Both editions (Standard and Datacenter) have the exact same functionality. The only difference between the two editions is the virtualization rights. Windows Server 2012 Standard edition gives the purchaser the rights to run 2 virtual instances of Windows Server, while the Datacenter Edition has unlimited virtualization rights.

13.3.2 Hyper-V Server

Hyper-V Server is a separate standalone product that is a free download. Hyper-V Server is a Hypervisor-based, meaning you don’t need to install it on top of an existing Windows Server.

Hyper-V Server only installs the Core Server features needed to run virtualization, you cannot other Roles like Active Directory, etc. and you don’t have a graphical interface. The normal way is to use Windows 8 and the Hyper-V Manager to manage the VMs remotely (create new VM, edit VMs, etc.).

The latest version is “Microsoft Hyper-V Server 2012 R2”.

13.3.3 Windows Client Hyper-V

In Windows 10 you can install the “Client Hyper-V” feature meaning you can run VMs inside Windows.

Step 1: Check if your PC are able to run Client Hyper-V

Check System Information (msinfo32.exe) to see if your PC is capable to run Windows Client Hyper-V (Figure 13-3):

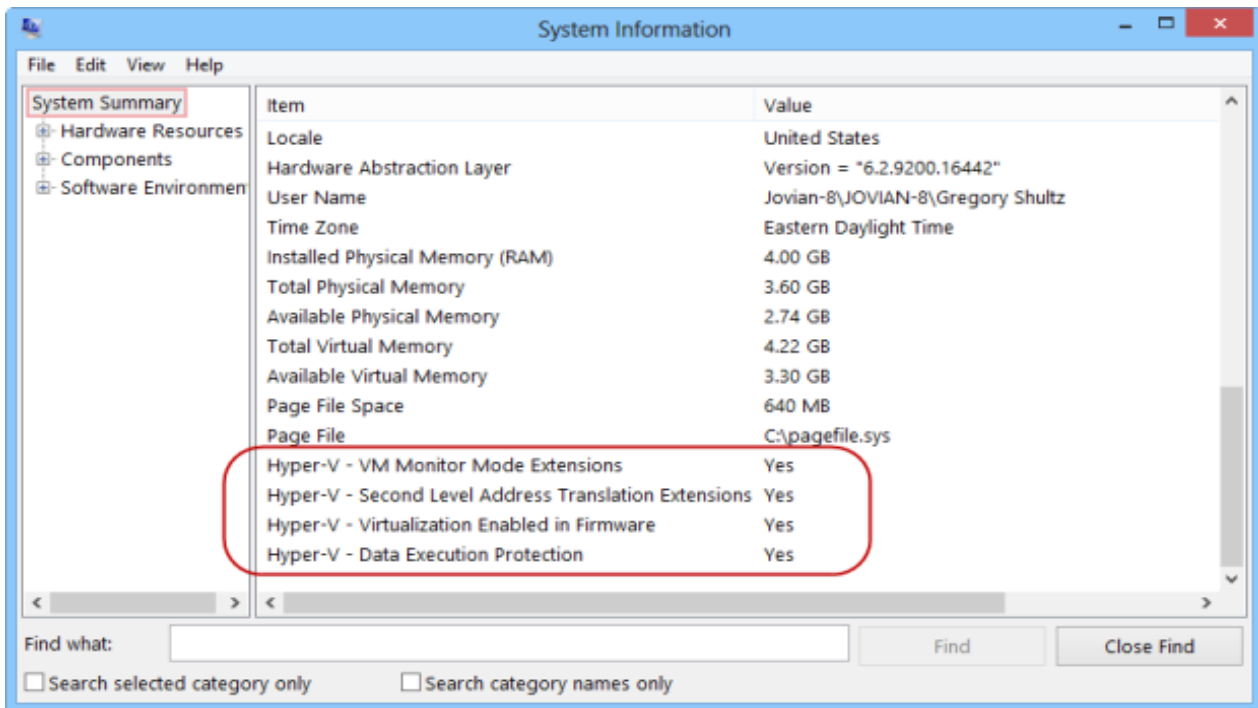


Figure 13-3: Hyper-V Requirements

All these settings need to be Yes. In general, if either the “Virtualization Enabled in Firmware” or the “VM Monitor Mode Extensions” are set to No, you can enable those features in the firmware. However, if the “Second Level Address Translation Extensions” or the “Data Execution Protection” settings are set to No, then you will not be able to use Windows 8 Client Hyper-V.

Step 2: Install Client Hyper-V

If your PC is able to run Client Hyper-V you need to add the Hyper-V feature in Windows (Figure 13-4).

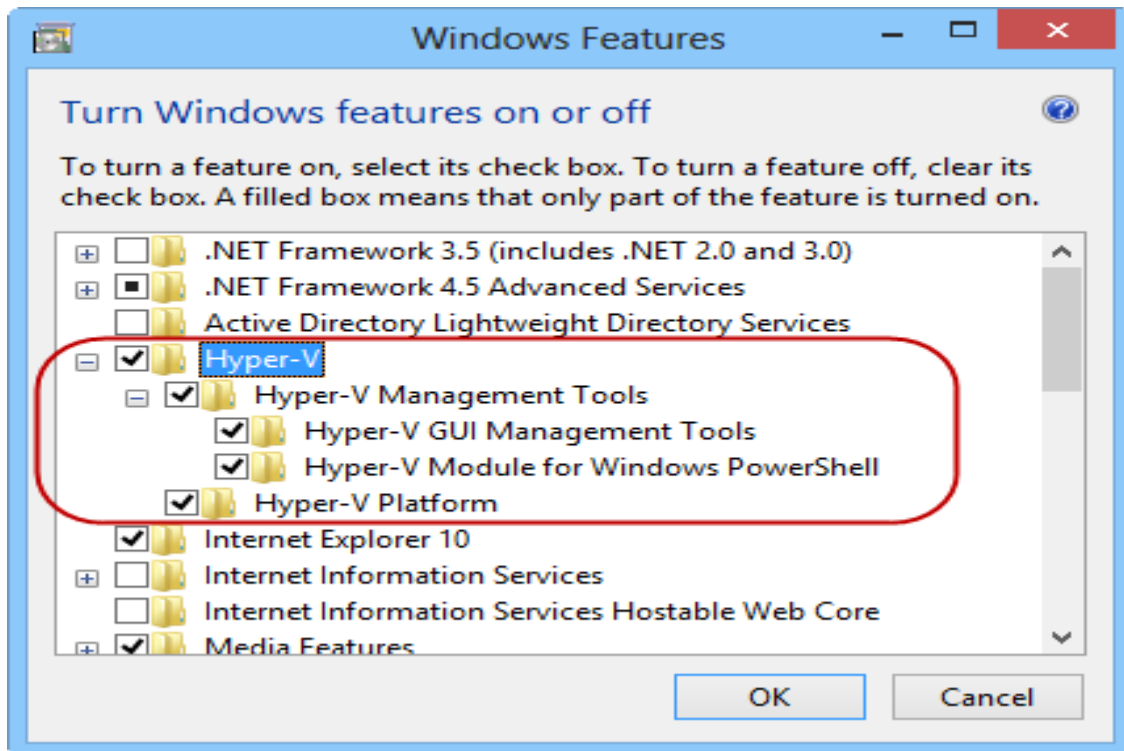


Figure 13-4: Hyper-V Installation Components

Then you can start up the Hyper-V Manager and start creating your VMs.

13.4 VirtualBox

VirtualBox is originally created by Sun Microsystems, but is now maintained by Oracle.

VirtualBox is freely available as Open Source Software.

Web site: <https://www.virtualbox.org>

VirtualBox is available for Windows, Mac OS X and Linux/UNIX.

13.5 Mac and OS X

If you have a Mac and want to run Windows or other OS, you have different options here as well.

- Boot Camp
- VMware Fusion
- Parallels Desktop
- VirtualBox

13.5.1 Boot Camp

Boot Camp is built into the Mac OS X. It is actually not a virtualization technique, but rather a method to run Windows on a Mac computer.

13.5.2 VMware Fusion

Web Site:

<http://www.vmware.com/products/fusion/overview.html>

Example (Figure 13-5):

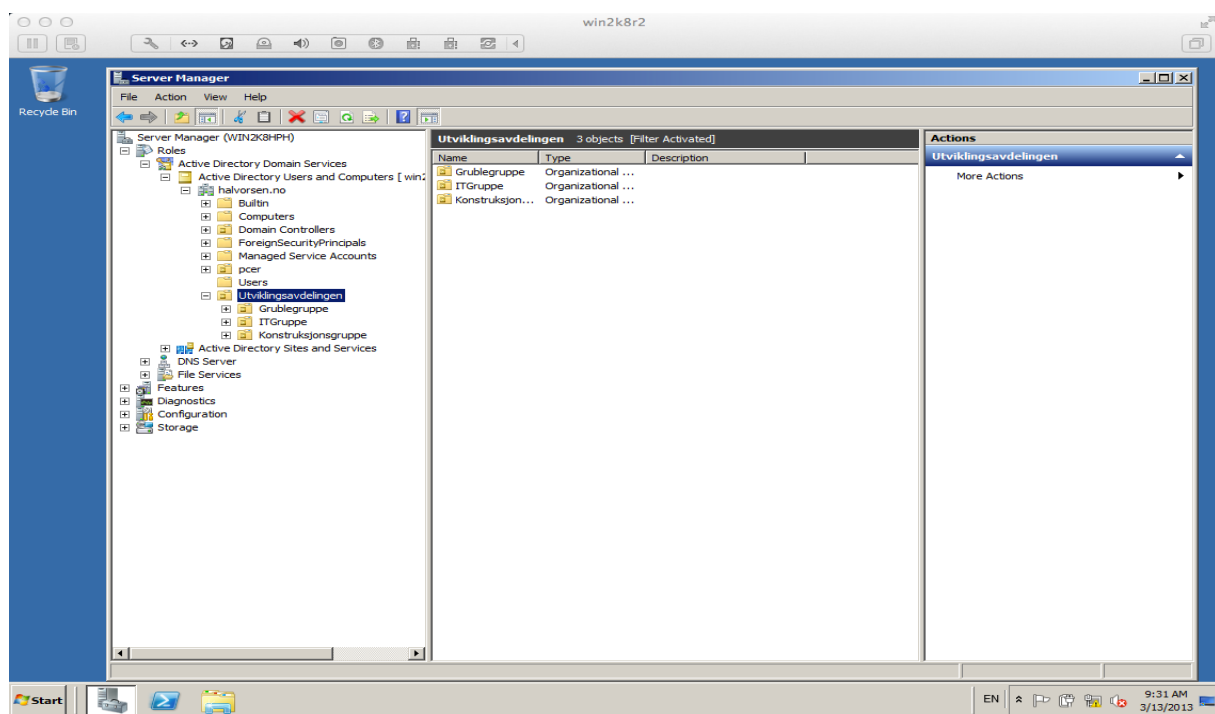


Figure 13-5: VMware Fusion on Mac OS X

13.5.3 Parallels Desktop

It costs about \$80. You can buy and download from the vendor's web site:

<http://www.parallels.com/products/desktop/>

13.5.4 VirtualBox

VirtualBox is originally created by Sun Microsystems, but is now maintained by Oracle.

VirtualBox is freely available as Open Source Software.

Web site: <https://www.virtualbox.org>

VirtualBox is available for Windows, Mac OS X and Linux/UNIX.

14 Cloud Computing

Web: <https://www.halvorsen.blog/documents/technology/cloud/>

Cloud computing, also known as on-demand computing, is a kind of internet-based computing, where shared resources and information are provided to computers and other devices on-demand (Wikipedia).

Figure 14-1 is showing overview of cloud computing, with typical types of applications supported by that computing model.

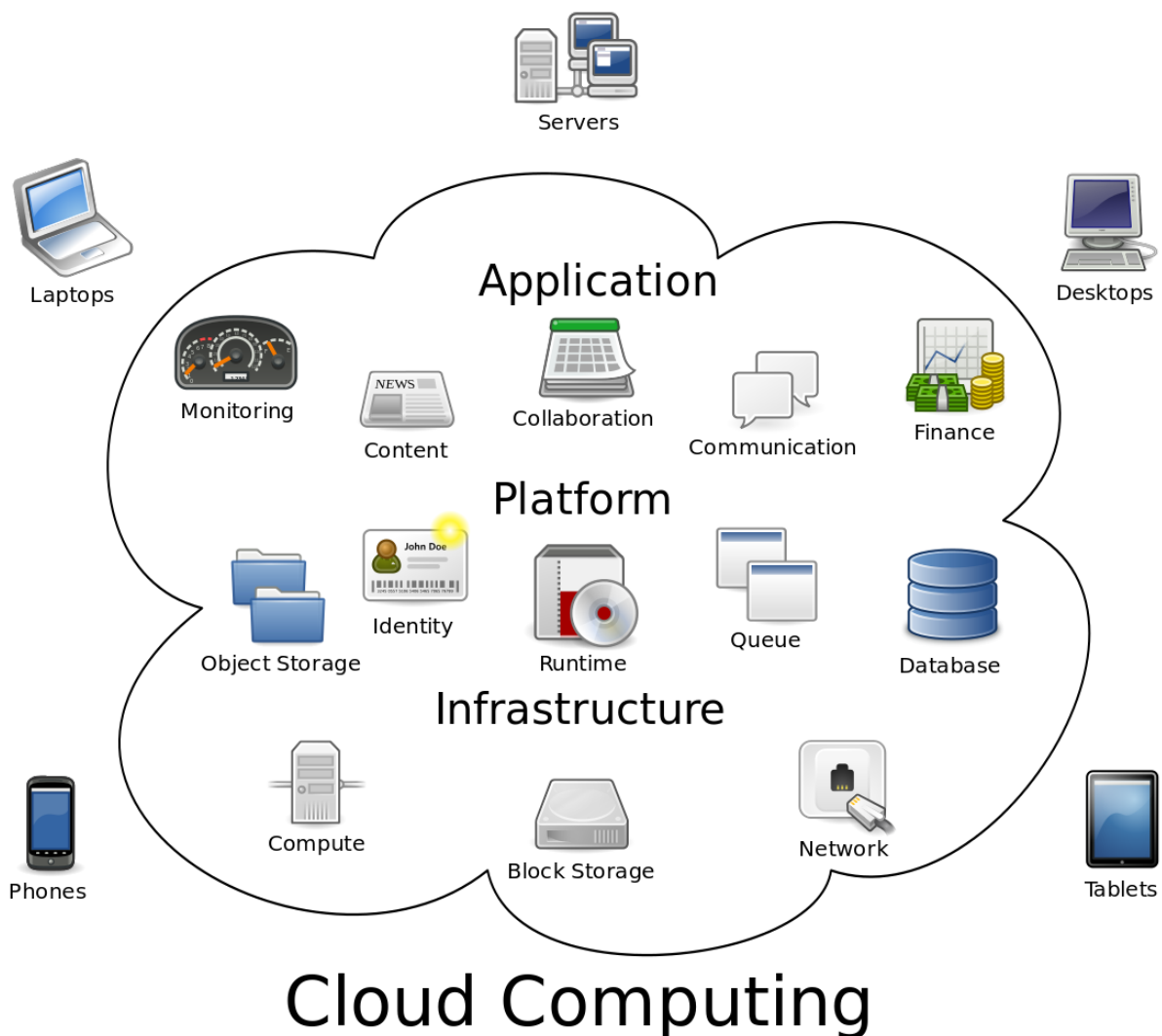


Figure 14-1: Cloud Computing (https://en.wikipedia.org/wiki/Cloud_computing, Sam Johnston)

Microsoft, Amazon and Google are 3 big players in this market with their respective products:

- Windows Azure
- Amazon Web Services
- Google Cloud Platform

Cloud Computing provides a simple way to access servers, storage, databases and a broad set of application services over the Internet. Cloud Computing providers such as Microsoft, Amazon, Google, etc. own and maintain the network-connected hardware required for these application services, while you provision and use what you need via a web application.

14.1 Microsoft Azure

Microsoft Azure is the cloud computing solution from Microsoft, you could say it is “Windows in the Cloud” with no local hardware/software installations, i.e. there is no need for local servers.

Cloud Features:

- Web Sites
- SQL Servers
- Virtual Machines
- etc.

The product is based on monthly payments (1-month free trial)

Windows Azure is available from here:

<https://azure.microsoft.com>

14.2 Amazon Web Services

Amazon Web Services (AWS) is another cloud service from Amazon.



Cloud Features:

- Web sites
- SQL Servers
- Virtual Machines

- etc.

Amazon Web Services (AWS) is available from here:

<http://aws.amazon.com>

“AWS Free Usage Tier” (available for 12 months):

<http://aws.amazon.com/free>

14.3 Google Cloud Platform

Google Cloud Platform is a cloud computing platform by Google that offers hosting on the same supporting infrastructure that Google uses internally for end-user products like Google Search and YouTube. Cloud Platform provides developer products to build a range of programs from simple websites to complex applications. https://en.wikipedia.org/wiki/Google_Cloud_Platform

Google Cloud Platform is available from here:

<https://cloud.google.com>

15 Wireless Systems

It's not just phones and computers that can communicate wirelessly with the outside world. Wireless sensors appear both in medicine, in buildings and especially in industrial applications.

Wireless technology and wireless networks are widely used today, but it's still quite new in industrial automation systems.

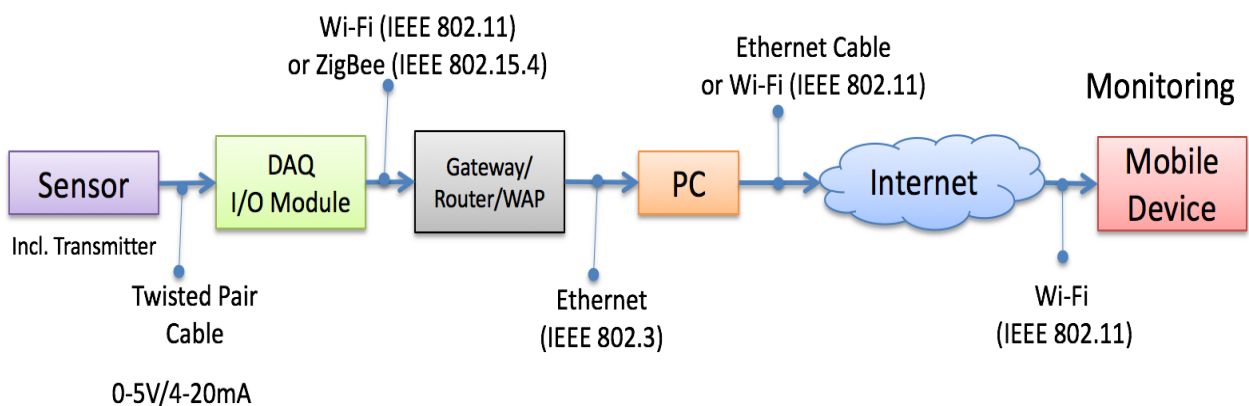


Figure 15-1: Wireless DAQ System

Here are some examples of Wireless Technologies:

- Cellular
- Bluetooth
- ZigBee (IEEE 802.15.4)
- Wireless USB
- Wi-Fi (IEEE 802.11)
- WirelessHART

ZigBee and WirelessHART are popular in industrial applications, while Wi-Fi, Bluetooth and Cellular are more popular in consumer products like Smartphones, computers, etc.

For more information about Wireless DAQ Systems, please read the Tutorial "Wireless Data Acquisition in LabVIEW".

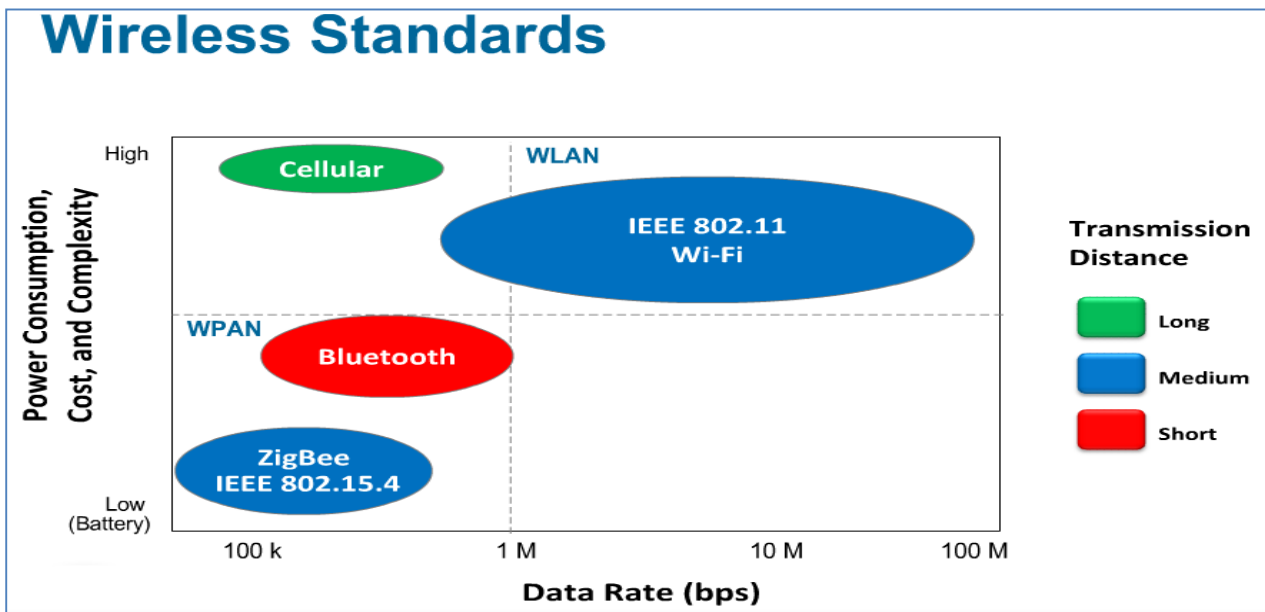


Figure 15-2: Overview of Wireless Standards

15.1 ZigBee

ZigBee (IEEE 802.15.4) is a low-cost, low-power, wireless mesh networking proprietary standard. The low cost allows the technology to be widely deployed in wireless control and monitoring applications, the low power-usage allows longer life with smaller batteries.

Read more about ZigBee:

ZigBee Alliance: <http://www.zigbee.org/>

Wikipedia: <http://en.wikipedia.org/wiki/ZigBee>

15.2 WirelessHART

WirelessHART is an open-standard wireless networking technology developed by HART Communication Foundation. It was developed as a multi-vendor, interoperable wireless standard.

Read more about WirelessHART:

HART Communication Foundation: <http://www.hartcomm.org/>

Wikipedia: <http://en.wikipedia.org/wiki/WirelessHART>

16 Vision Systems

Vision systems are important in the industry (and in other areas) today. Figure 16-1 shows a simple vision system where a vision system is used to remove bad items.

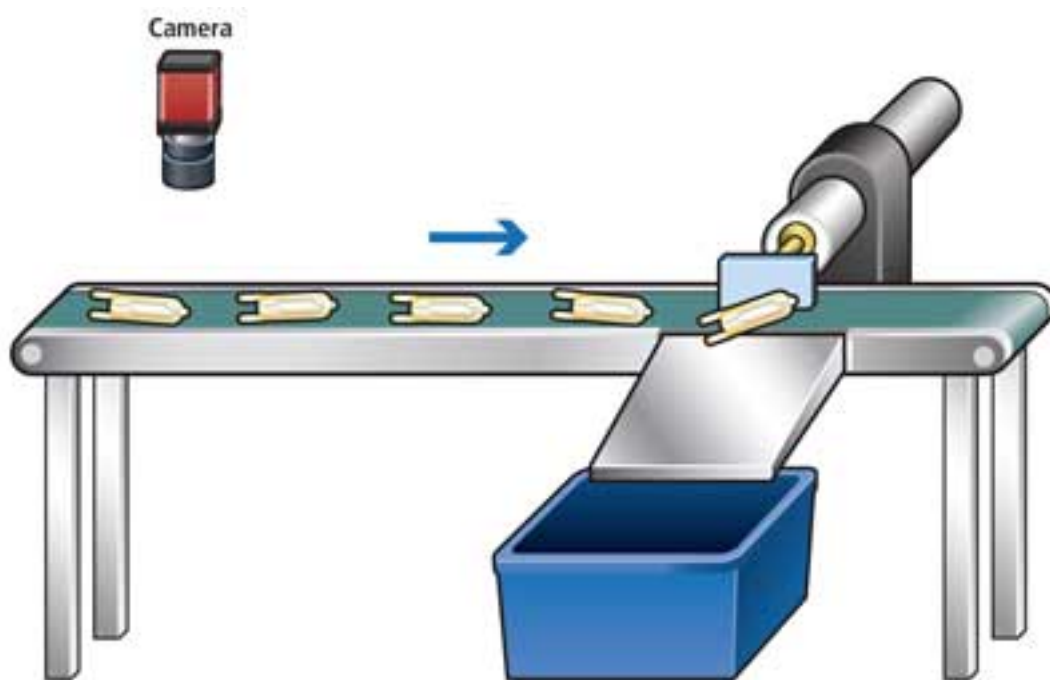


Figure 16-1: Simple Vision System

16.1 Vision Systems in LabVIEW

With LabVIEW, you can create advanced vision systems. You need the following software:

- LabVIEW
- NI Vision Acquisition Software
- NI Vision Development Module

The **NI Vision Acquisition software** is the basic software you need if you want to create Vision applications for LabVIEW or the .NET platform. The NI Vision Acquisition software includes the necessary drivers, such as NI-IMAQ and NI-IMAQdx.

The **NI-IMAQdx** driver software gives you the ability to acquire images with IEEE 1394 (FireWire), GigE Vision (Ethernet), and USB cameras.

For more advanced machine vision and image processing you will need the **Vision Development Module**. The Vision Development Module contains hundreds of image-processing and machine vision functions, both for LabVIEW and the .NET platform.

This package includes built-in functions for:

- Pattern matching
- Texture recognition
- Counting and Classification
- OCR (Optical Character Recognition)
- Bar Code readers
- Image Filters
- etc.

Figure 16-2 shows the LabVIEW Vision palette.

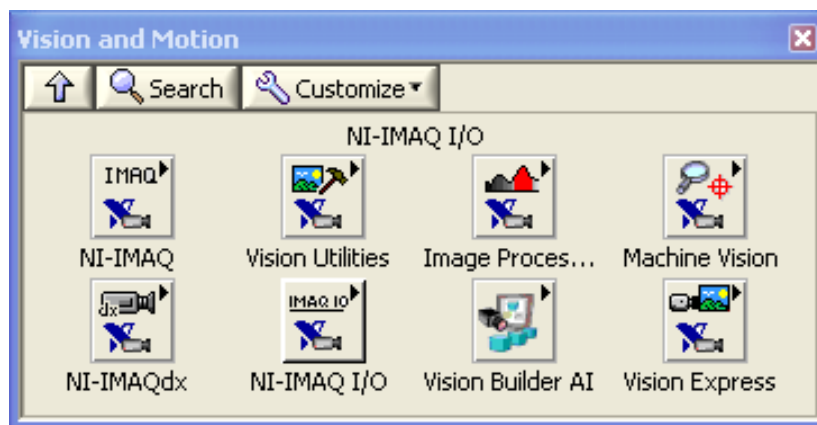


Figure 16-2: LabVIEW Vision Palette

16.2 Vision Systems in Visual Studio/C#

The **NI Vision Acquisition software** is the basic software you need if you want to create Vision applications for LabVIEW or the .NET platform. The NI Vision Acquisition software includes the necessary drivers, such as NI-IMAQ and NI-IMAQdx.

The **NI-IMAQdx** driver software gives you the ability to acquire images with IEEE 1394 (FireWire), GigE Vision (Ethernet), and USB cameras.

Part 3 : Automation

In this part we give an overview of DAQ systems, control systems, sensors and actuators, OPC, SCADA systems, Hardware-in-the-Loop simulations, etc.

17 DAQ Systems

Web: <https://www.halvorsen.blog/documents/technology/daq/>

DAQ is short for Data Acquisition. The purpose of data acquisition is to measure an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound. PC-based data acquisition uses a combination of modular hardware, application software, and a computer to take measurements. While each data acquisition system is defined by its application requirements, every system shares a common goal of acquiring, analyzing, and presenting information. Data acquisition systems incorporate signals, sensors, actuators, signal conditioning, data acquisition devices, and application software.

So summing up, Data Acquisition is the process of:

- Acquiring signals from real-world phenomena
- Digitizing the signals
- Analyzing, presenting and saving the data

The DAQ system has the following parts involved, see Figure 17-1.

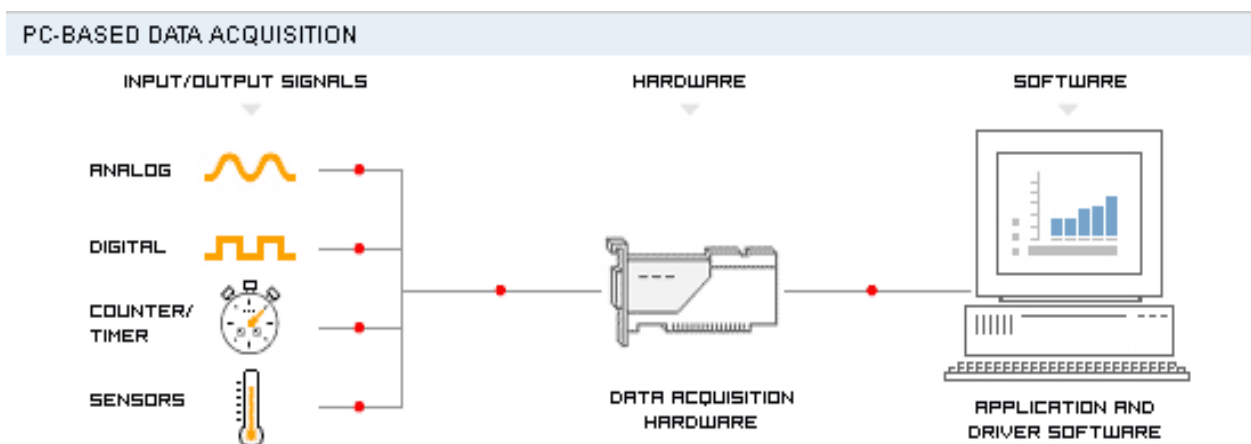


Figure 17-1: DAQ System

The parts in a typical DAQ system are as follows:

- Physical input/output signals
- DAQ device/hardware
- Driver software
- Your software application (Application software)

17.1 Sampling

Discrete sampling of a continuous signal:

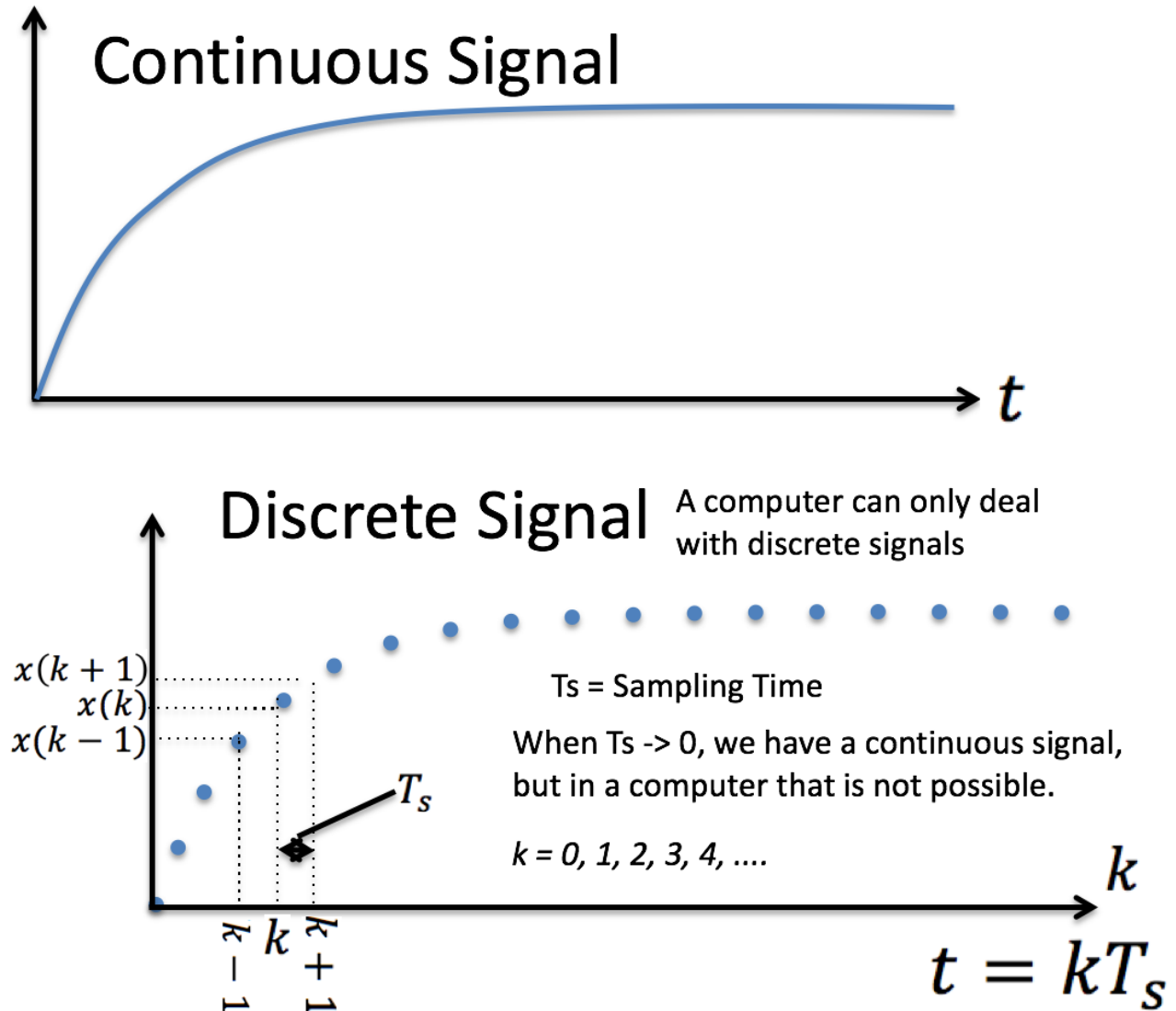


Figure 17-2: Continuous vs. Discrete Signals

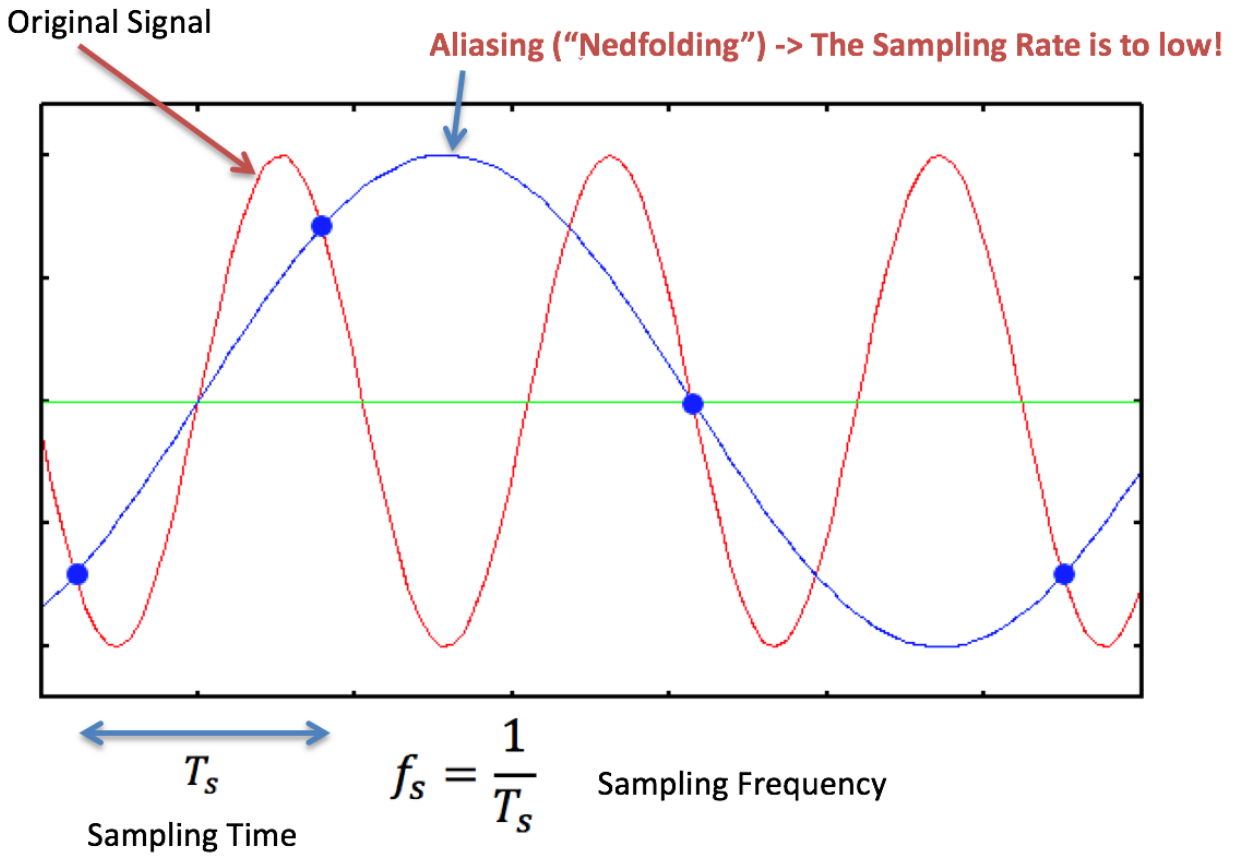


Figure 17-3: Aliasing

17.1.1 AD Converters

All Analog Signals needs to be converted to Digital Signals before the Computer can use them (AD Converter).

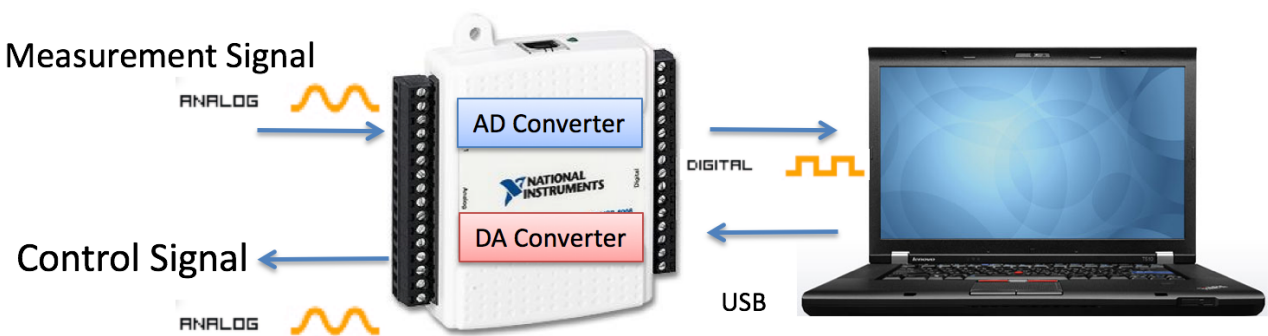


Figure 17-4: Analog to Digital Converter

AD – Analog to Digital

DA – Digital to Analog

What is an Analog-to-Digital Converter (ADC)?

Analog signals from sensors must be converted into digital before they are manipulated by digital equipment such as a computer. An ADC is a chip that provides a digital representation of an analog signal at an instant in time. In practice, analog signals continuously vary over time and an ADC takes periodic “samples” of the signal at a predefined rate. These samples are transferred to a computer over a computer bus where the original signal is reconstructed from the samples in software.

17.2 DAQ Hardware

What Is a DAQ Device?

DAQ hardware acts as the interface between a computer and signals from the outside world. It primarily functions as a device that digitizes incoming analog signals so that a computer can interpret them. The three key components of a DAQ device used for measuring a signal are the signal conditioning circuitry, analog-to-digital converter (ADC), and computer bus.

In this document, we will use different hardware available from National Instruments in our examples, these devices are:

- TC-01 Thermocouple device
- USB-6008 DAQ Device

17.2.1 NI USB TC-01 Thermocouple Device

In Figure 17-5 we see the NI USB-TC01 Thermocouple Measurement device.



Figure 17-5: TC-01 Thermocouple Device

We will give code examples of how to use this device in C#. Since this is a DAQmx supported device from National Instruments, the programming structure will be the same as for NI USB-6008.

17.2.2 NI USB-6008 DAQ Device

NI USB-6008 (see Figure 17-6) is a simple and low-cost multifunction I/O device from National Instruments.



Figure 17-6: USB-6008 I/O Module

The device has the following specifications:

- 8 analog inputs (12-bit, 10 kS/s)
- 2 analog outputs (12-bit, 150 S/s)
- 12 digital I/O
- USB connection, no extra power-supply needed
- Compatible with LabVIEW and Visual Studio/C#
- NI-DAQmx driver software

The NI USB-6008 is well suited for education purposes due to its small size and easy USB connection.

With NI USB-6008 (or similar DAQ devices) you can connect all kinds of sensors, as long as it is a analog voltage signal. See Figure 17-7.

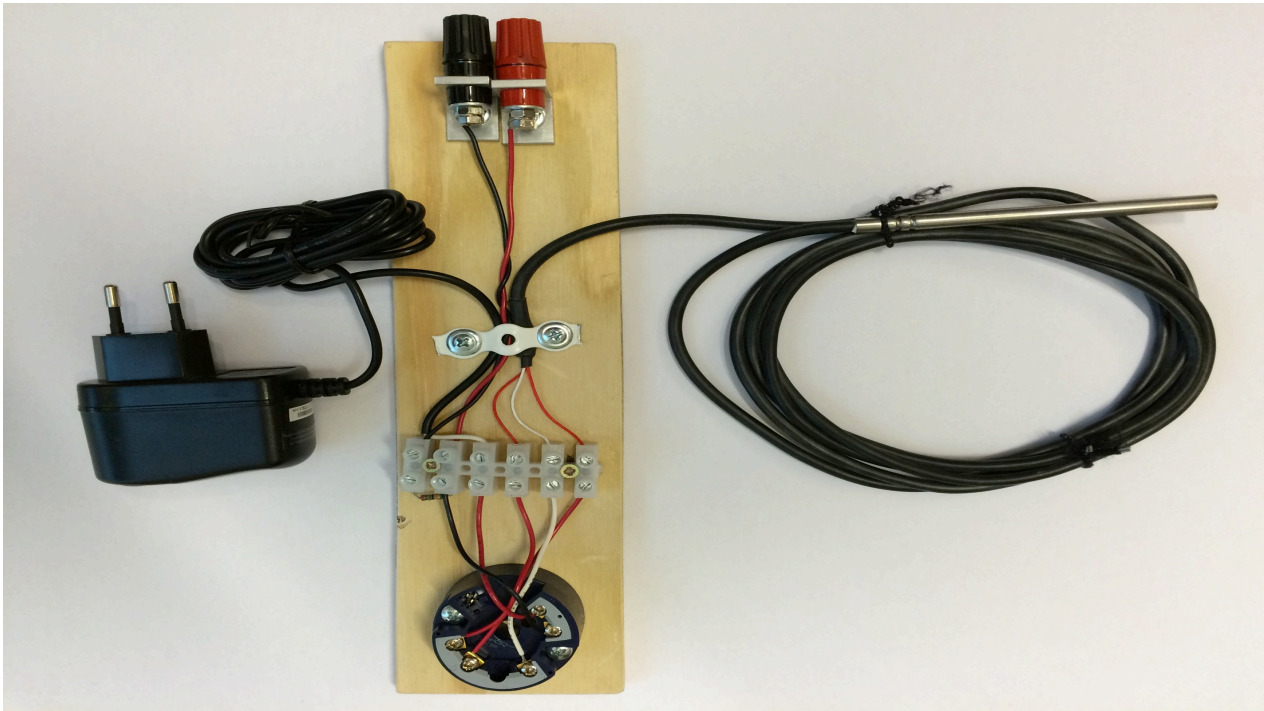


Figure 17-7: Pt-100 Measurements

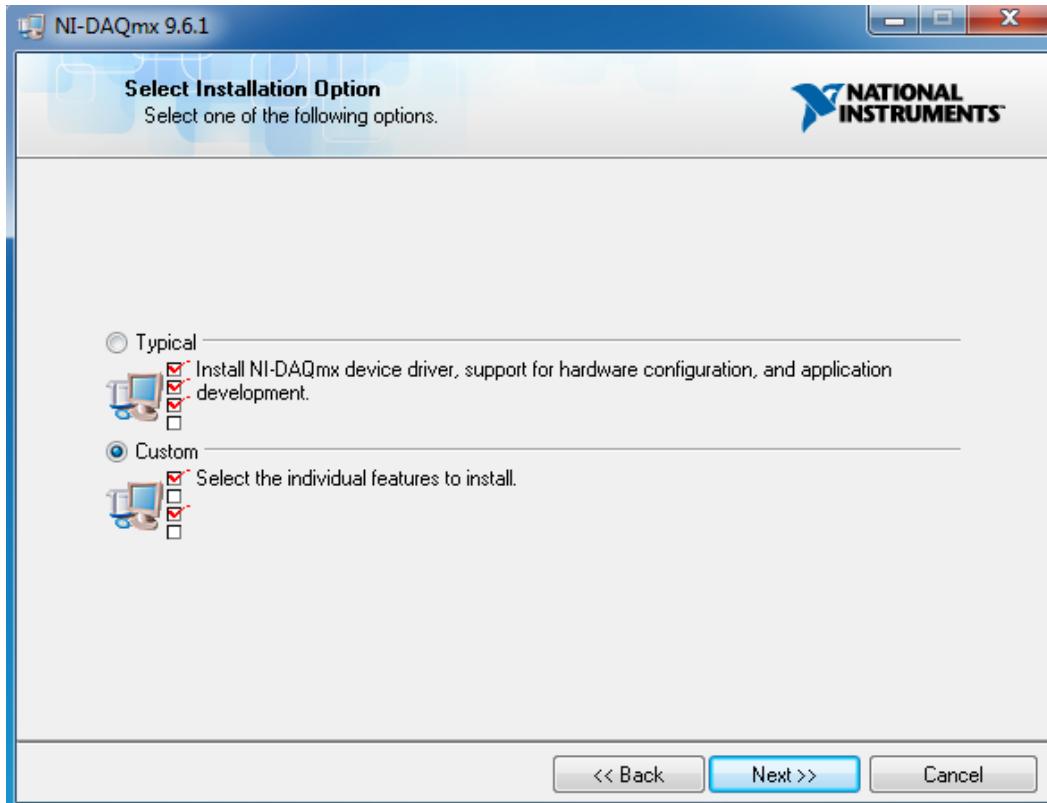
17.3 NI DAQmx driver

National Instruments provides a native .NET API for NI-DAQmx. This is available as a part of the NI-DAQmx driver and does not require Measurement Studio.

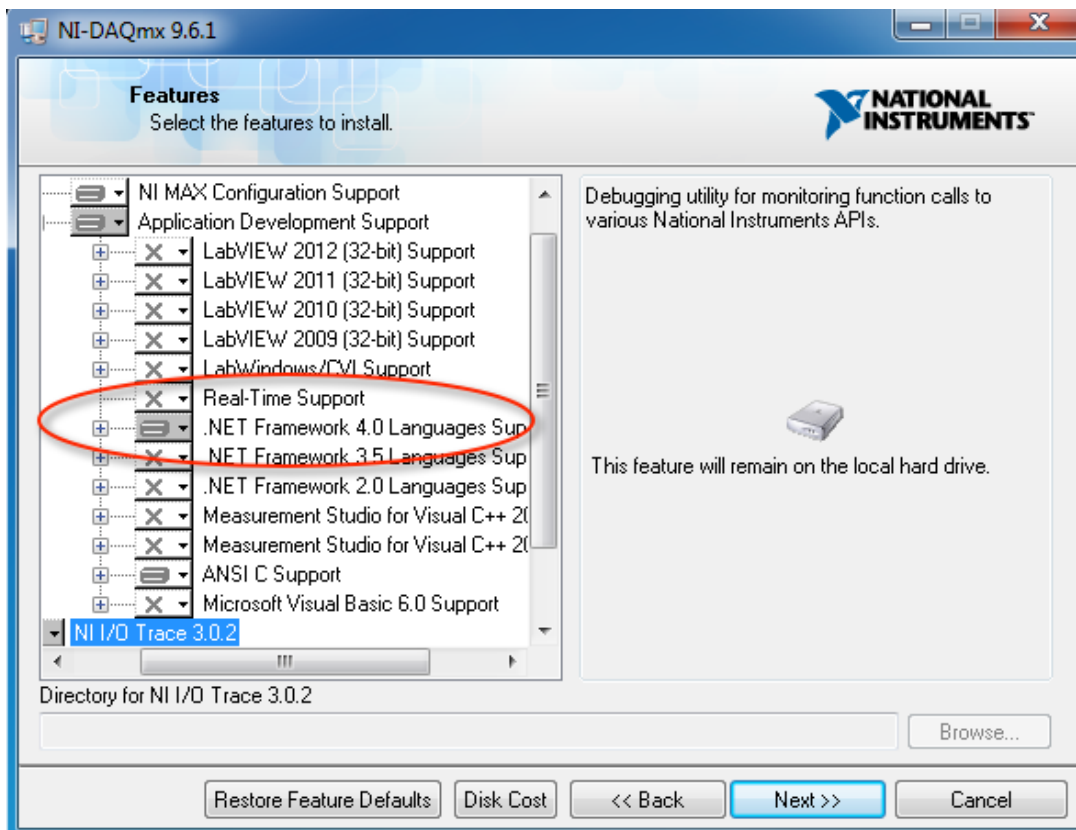
Note! In order to install the DAQmx API for C#, make sure to select “.NET Support” when installing the DAQmx driver.

This application uses the C# API included in the NI DAQmx driver, so make sure that you have installed the NI DAQmx driver in advance.

During the installation make sure to select “Custom”:



Next, make sure that you select .NET Framework X.x Support for the version of .NET that your version of Visual Studio is using:



17.3.1 NI MAX

Measurement & Automation Explorer (MAX) provides access to your National Instruments devices and systems.

With MAX, you can:

- Configure your National Instruments hardware and software
- Create and edit channels, tasks, interfaces, scales, and virtual instruments
- Execute system diagnostics
- View devices and instruments connected to your system
- Update your National Instruments software

In addition to the standard tools, MAX can expose item-specific tools you can use to configure, diagnose, or test your system, depending on which NI products you install. As you navigate through MAX, the contents of the application menu and toolbar change to reflect these new tools.

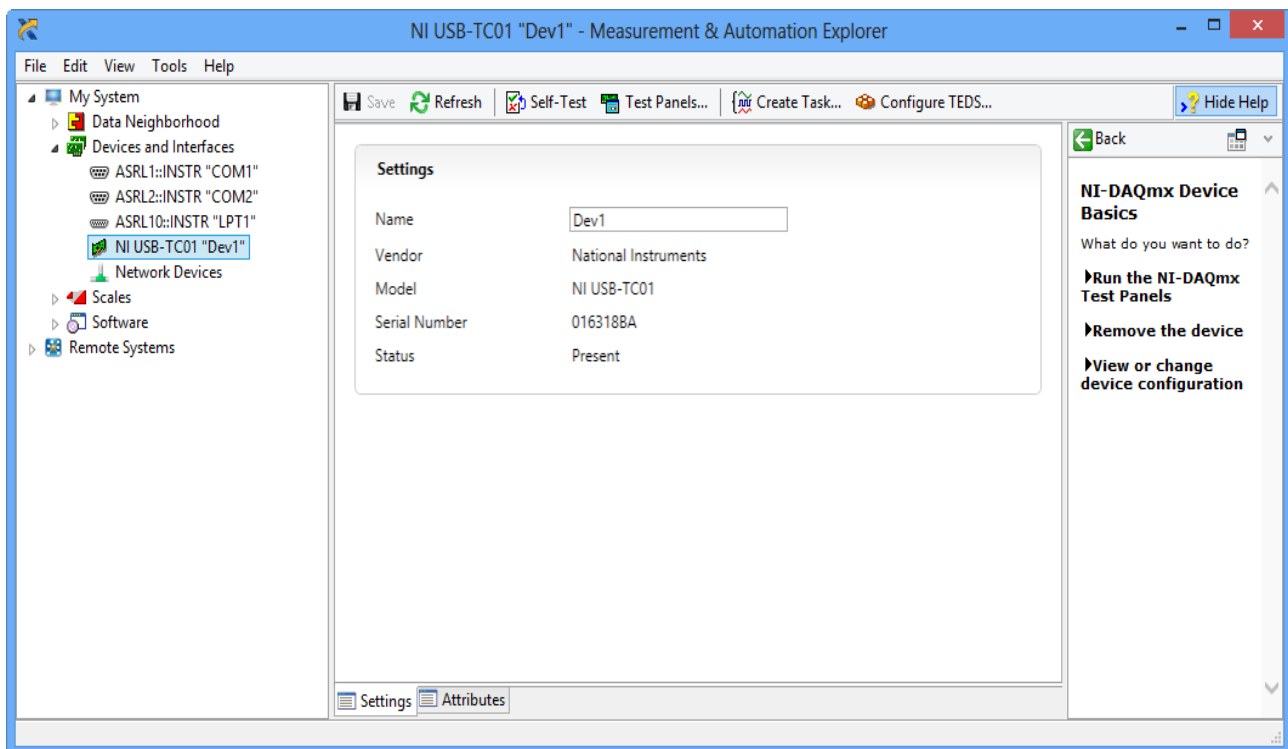


Figure 17-8: Measurement and Automation Explorer (MAX)

17.4 Measurement Studio

C# is a powerful programming language, but has few built-in features for measurement and control applications. Measurement Studio is an add-on to Visual Studio which makes it easier to

create such applications. With Measurement Studio, we can implement Data Acquisition and a graphical HMI.

17.5 DAQ with LabVIEW

After we have installed the DAQmx driver, we can easily use NI DAQ devices in LabVIEW. Figure 17-9 shows the DAQmx palette in LabVIEW. You find the palette in the Functions Palette: “Measurement I/O” -> “NI DAQmx”.

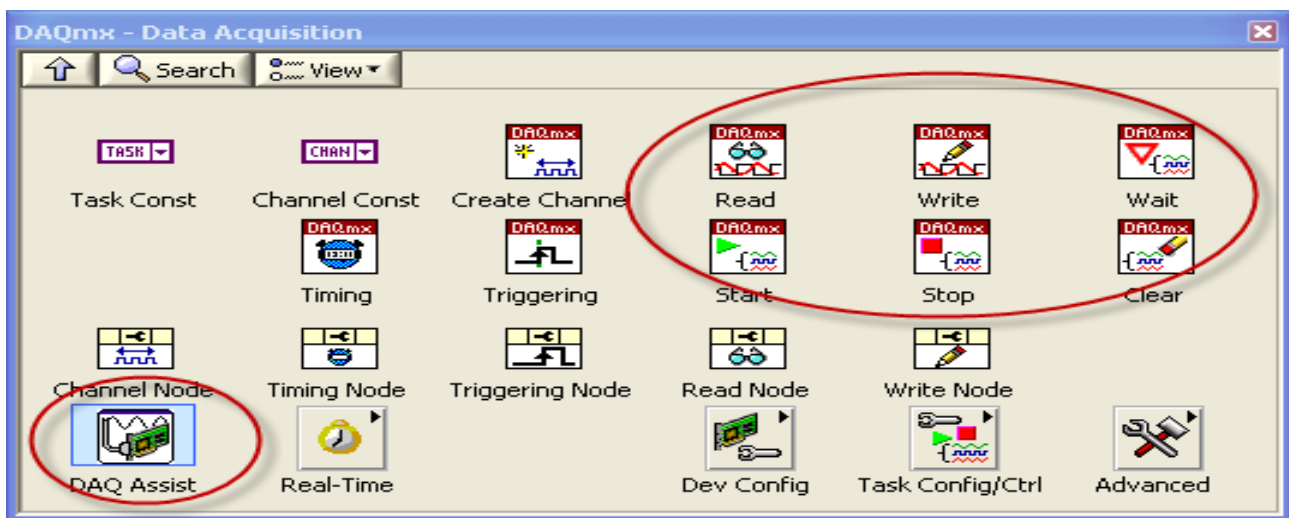


Figure 17-9: DAQmx Palette in LabVIEW

For basic DAQ we use the DAQ Assistant, for more “advanced” DAQ we use the Start/Stop and Read/Write functions.

When you place the DAQ Assistant (see Figure 17-10) on the Block Diagram, a Wizard automatically pops up where you configure what you want to do, i.e., if you want to Read or Write Data, Analog or Digital signals, which channel you want to use, etc.

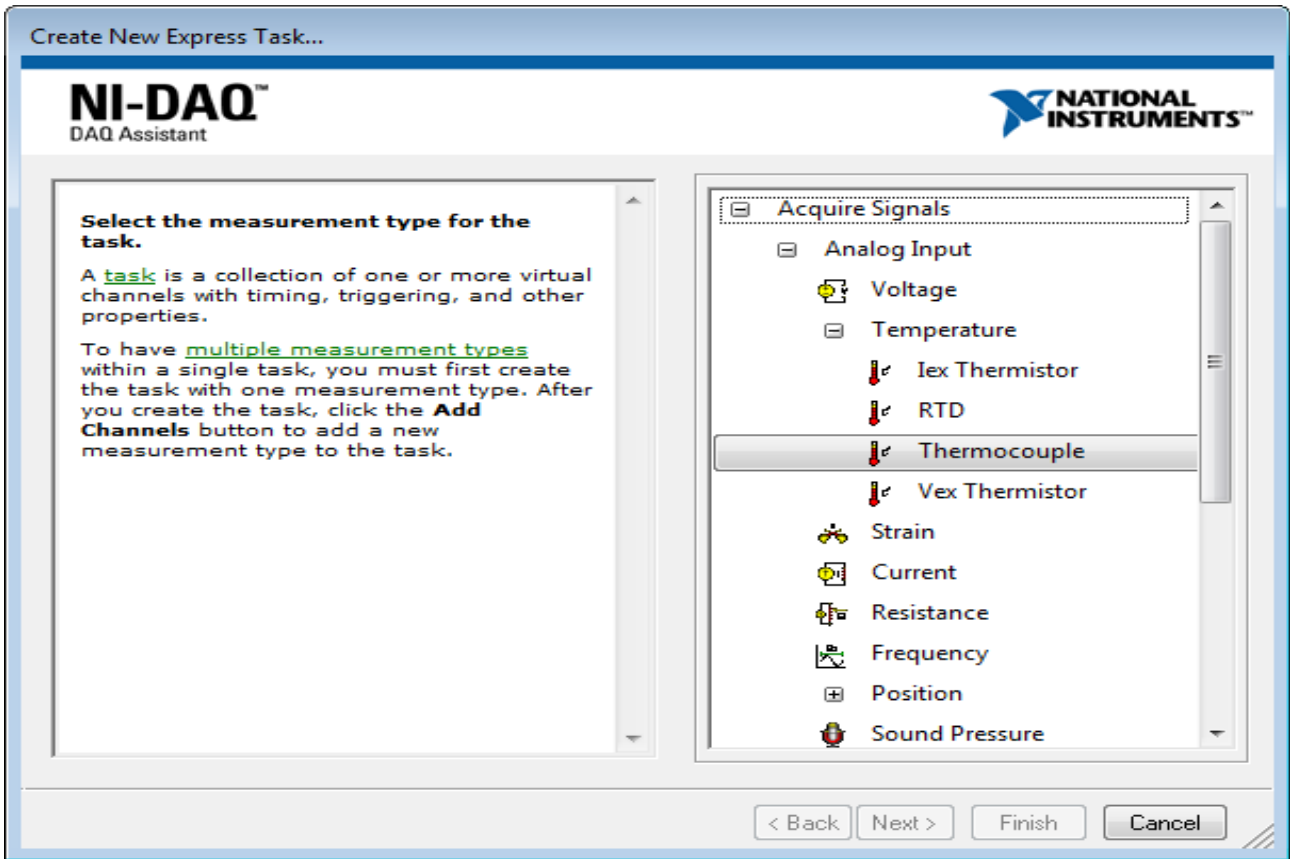


Figure 17-10: LabVIEW DAQ Assistant

The following steps needs to be taken in the DAQ Assistant (Figure 17-11):

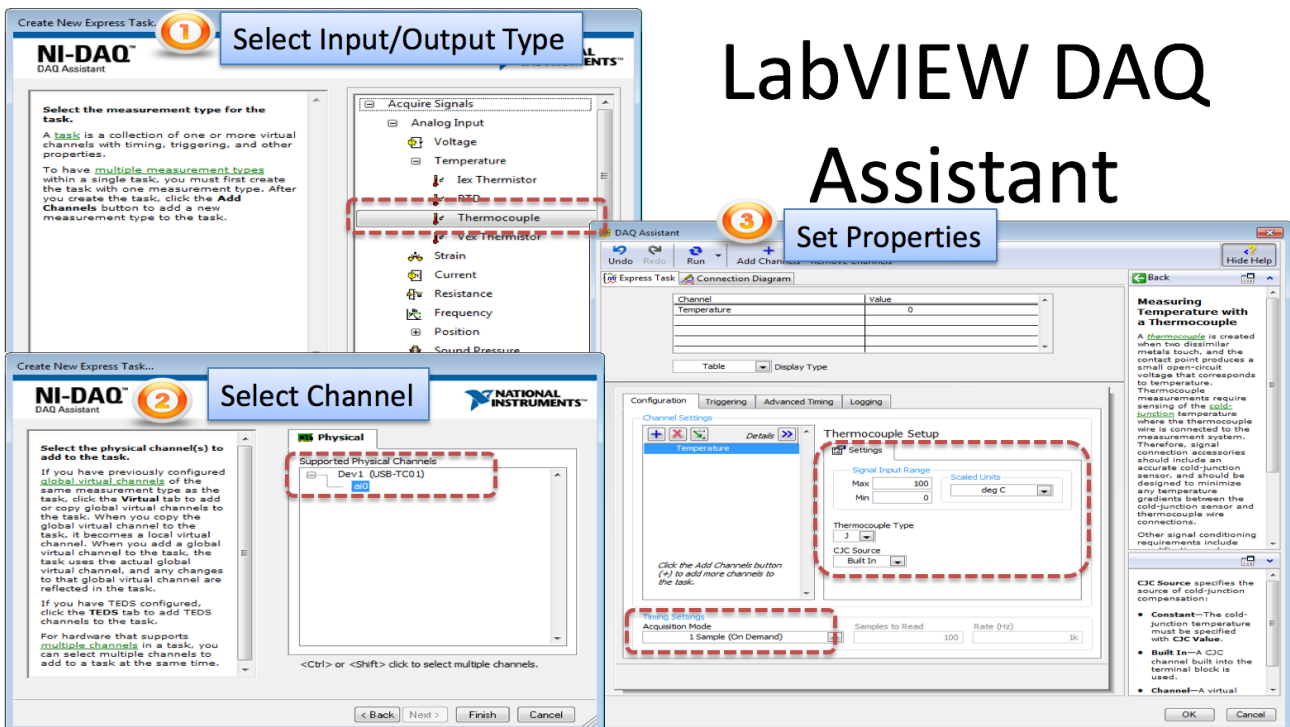


Figure 17-11: LabVIEW DAQ Assistant – Step by step

Figure 17-12 shows a simple DAQ Application in LabVIEW.

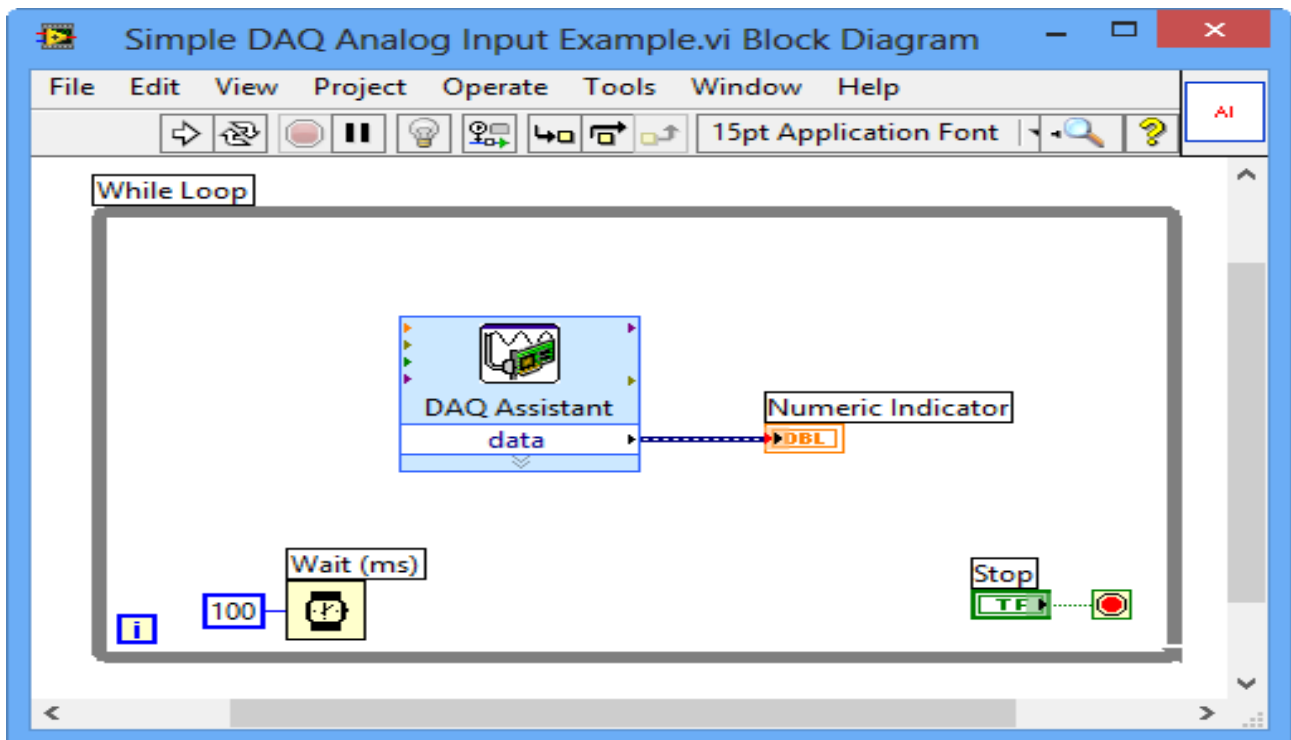


Figure 17-12: Simple DAQ Example in LabVIEW

17.5.1 Write to DAQ

In addition to read from the DAQ device we can also write to the DAQ device. See Figure 17-13

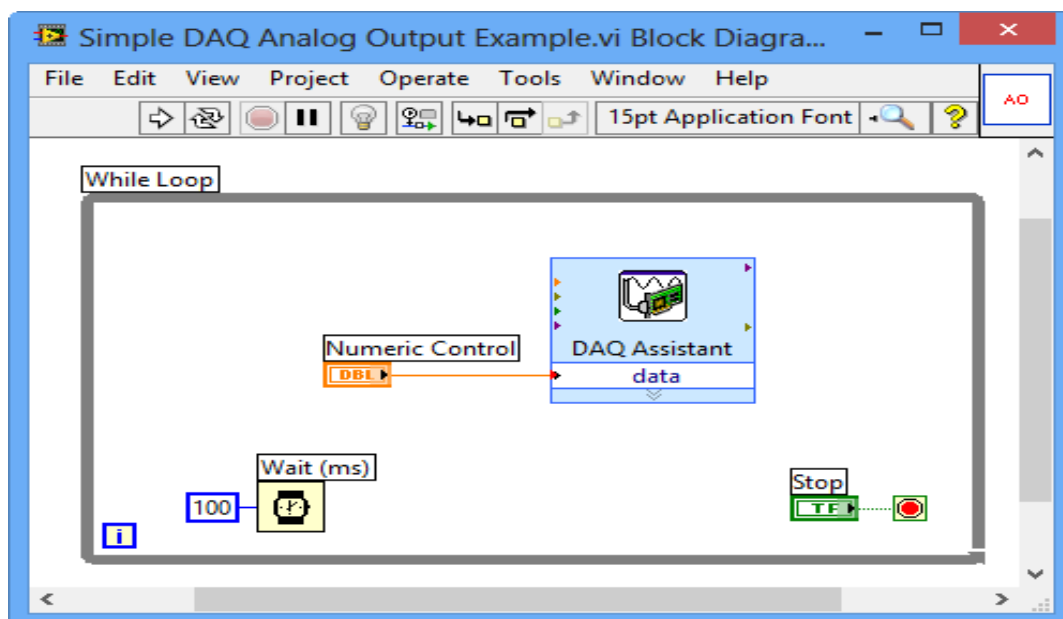


Figure 17-13: Write to DAQ (Analog Out) Example

Figure 17-14 shows the settings in the DAQ Assistant for Analog Out.

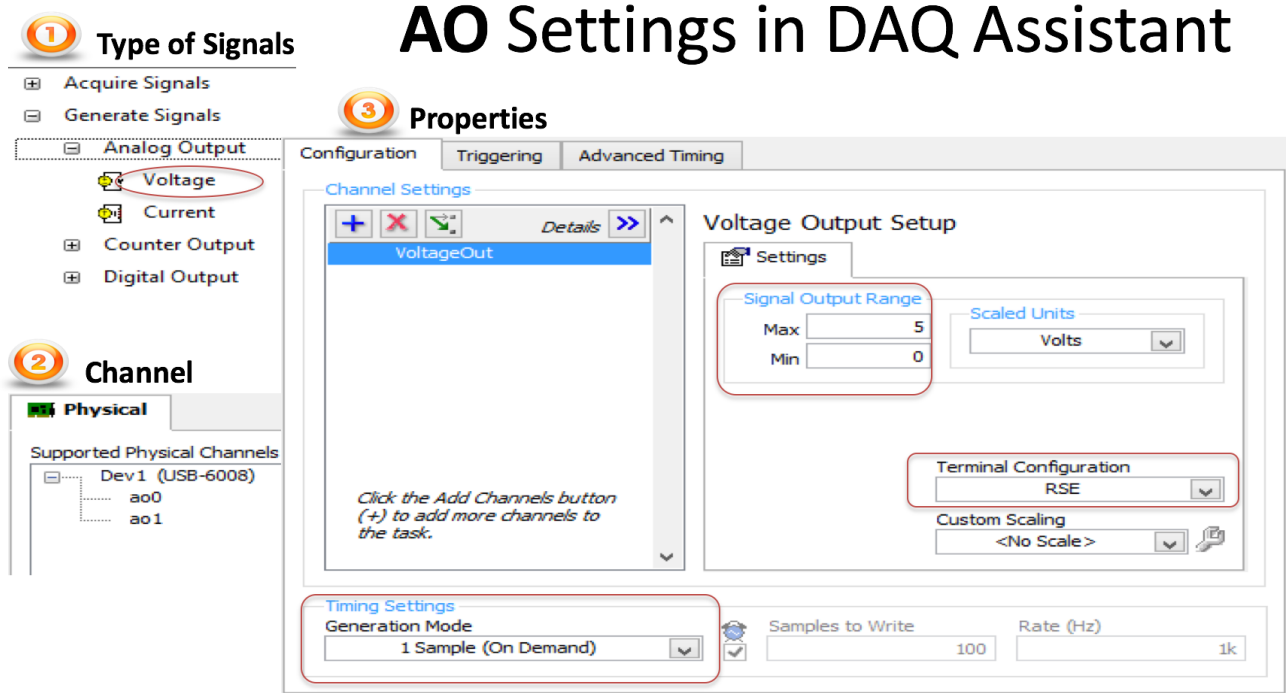


Figure 17-14: DAQ Assistant Settings for Analog Out

17.6 Datalogging

Figure 17-15 shows a simple Datalogging example implemented in LabVIEW. In this example we log data from a sensor and presenting the data into a chart. The application also saves the data to a file.

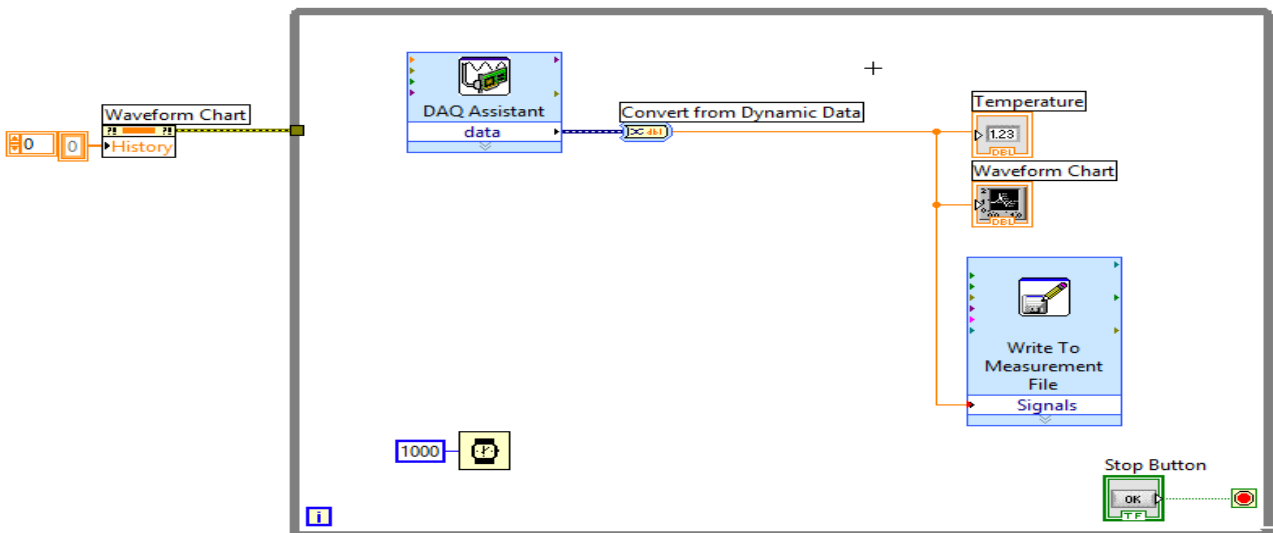


Figure 17-15: Simple Datalogging Example

Figure 17-16 shows the Front Panel (GUI/HMI) for the logging application.

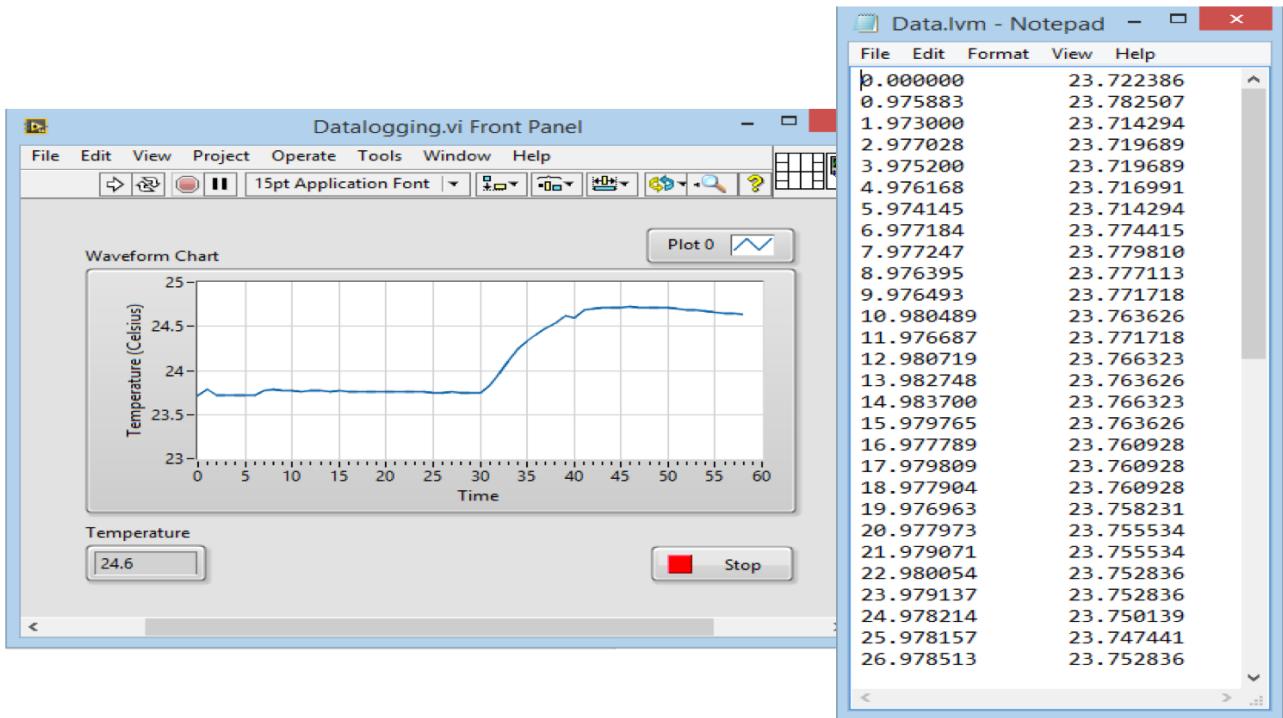
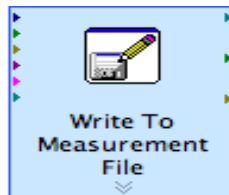


Figure 17-16: The GUI for the logging Application, including the Data stored on File

Saving the data to a so-called Measurement File in LabVIEW is very easy. You just use the “Write To Measurement File”



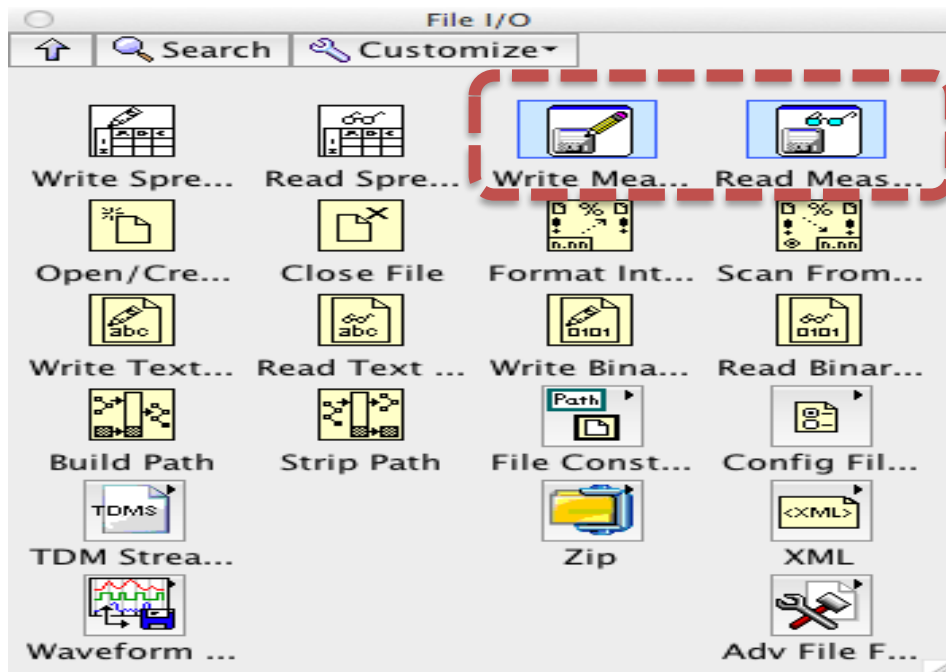


Figure 17-17: The File I/O palette in LabVIEW

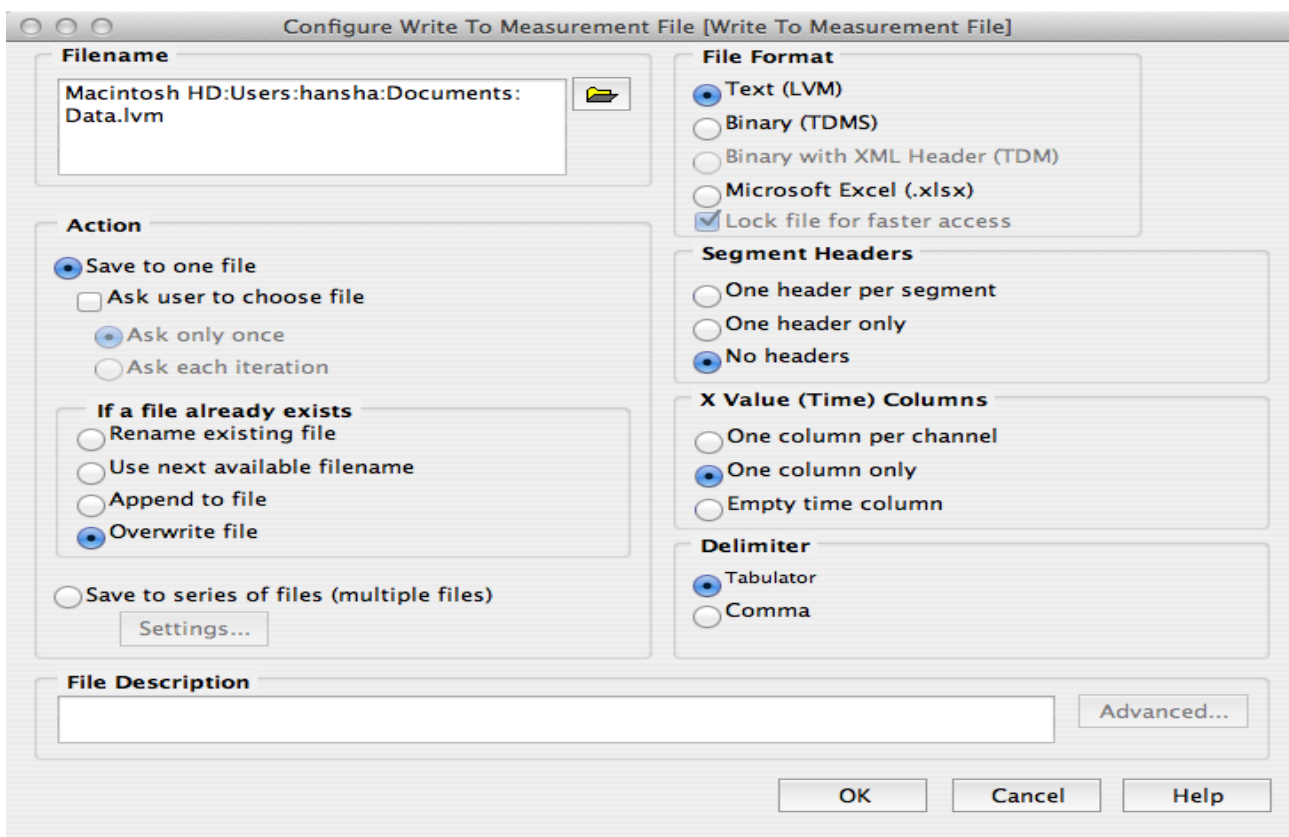


Figure 17-18: Configuration of the Log File Format

17.6.1 Measurement Filter/Low-pass Filter

The transfer function for a first-order low-pass filter may be written:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{T_f s + 1}$$

Where T_f is the time-constant of the filter, $u(s)$ is the filter input and $y(s)$ is the filter output.

Discrete version:

It can be shown that a discrete version can be stated as:

$$y_k = (1 - a)y_{k-1} + au_k$$

Where

$$a = \frac{T_s}{T_f + T_s}$$

Where T_s is the Sampling Time.

It is a golden rule that $T_s \ll T_f$ and in practice we should use the following rule:

$$T_s \leq \frac{T_f}{5}$$

17.6.2 LabVIEW Example

There are many ways to implement a Low-pass filter in LabVIEW. In this example, we will use the Formula Node in order to implement it from scratch.

Figure 17-19 shows the Front Panel with inputs and outputs for the Low-pass Filter we are going to create in LabVIEW.

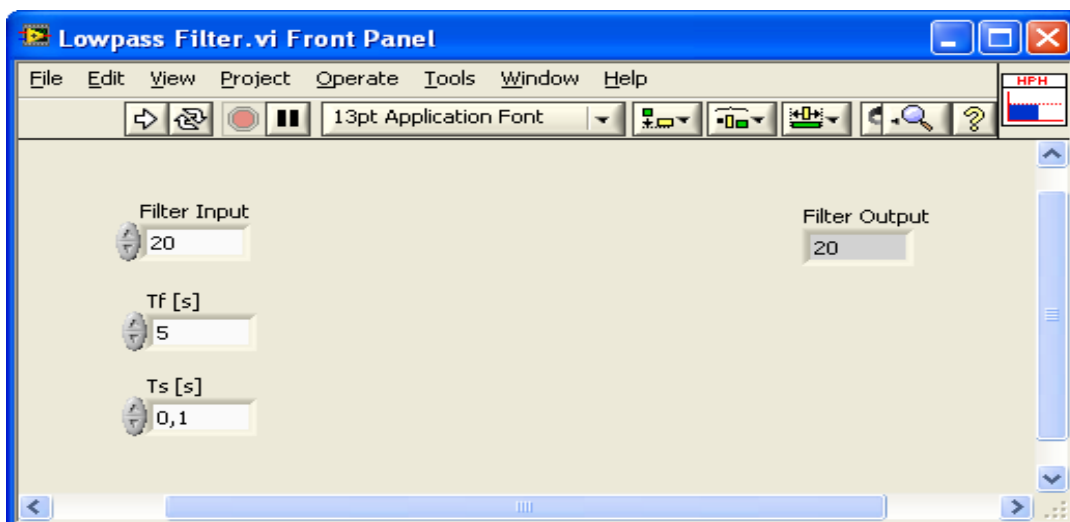


Figure 17-19: Low-pass Filter Front Panel

Figure 17-20 show the code for the Low-pass Filter implemented in LabVIEW.

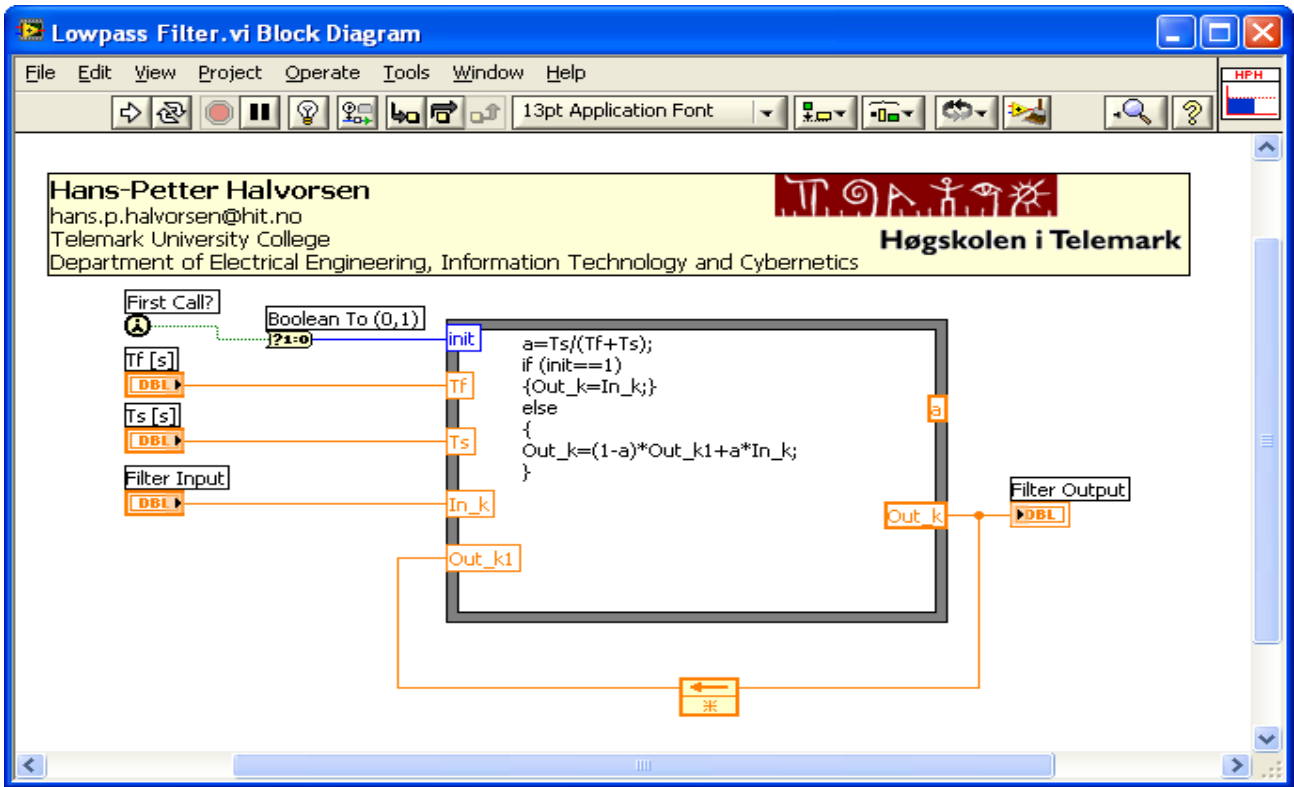


Figure 17-20: Low-pass Filter implemented in LabVIEW

Testing the Filter:

In this example (see Figure 17-21) we add noise to a Sine function. We then use the Measurement Filter to see if we can remove the noise afterwards.

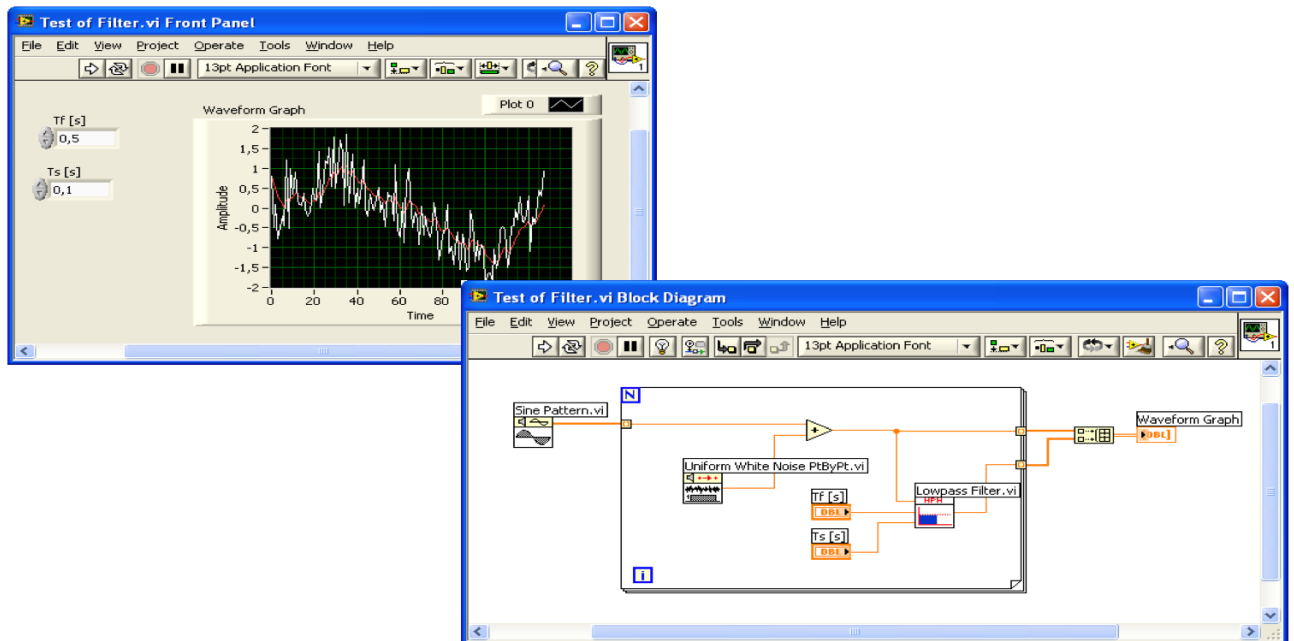


Figure 17-21: Testing the Low-pass Filter to make sure it works

As you can see this gives good results. The filter removes the noise from the signal.

We can also use the built-in “Filter” function in LabVIEW



You find the “Filter” function in the Functions palette: Express -> Signal Analysis -> Filter.

This function is more flexible, since you can configure it.

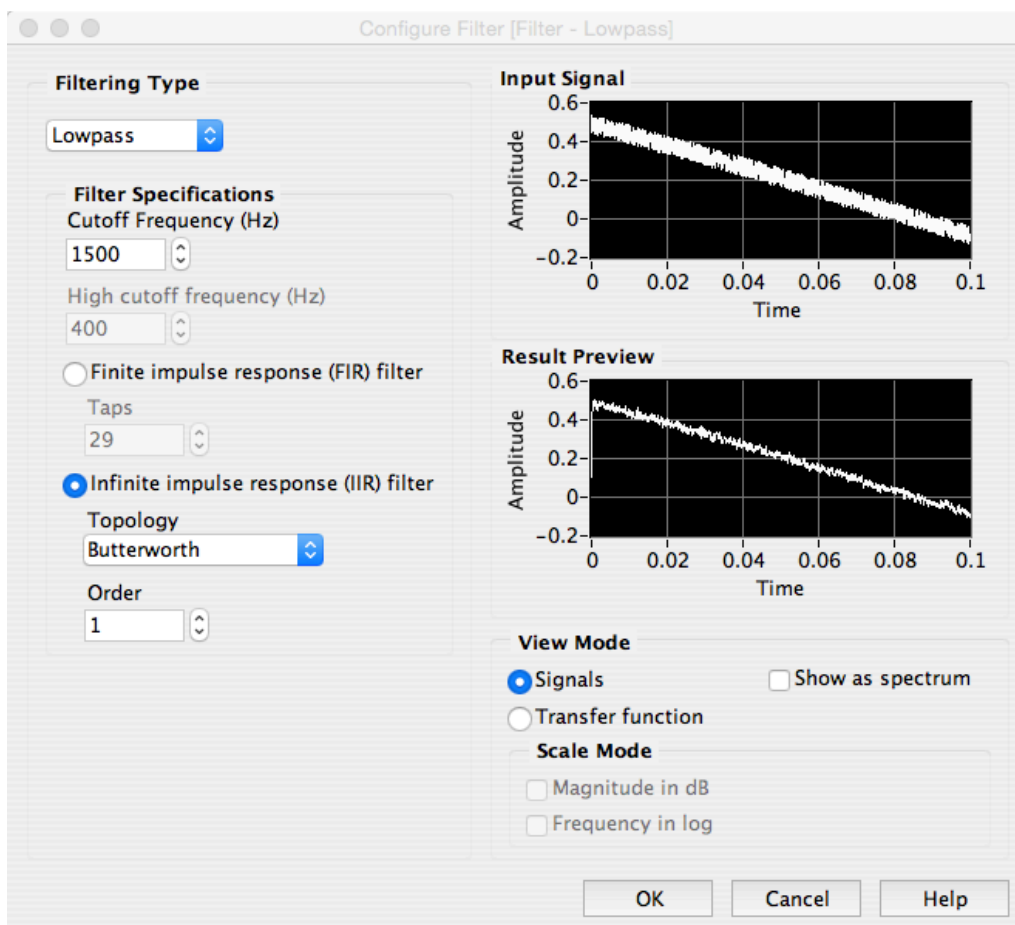


Figure 17-22: Built-in Filter in LabVIEW

Figure 17-23 shows a LabVIEW example using the built-in function in LabVIEW. In this example, we just simulate a signal with noise, and then we use the Filter function to see if we are able to remove the noise.

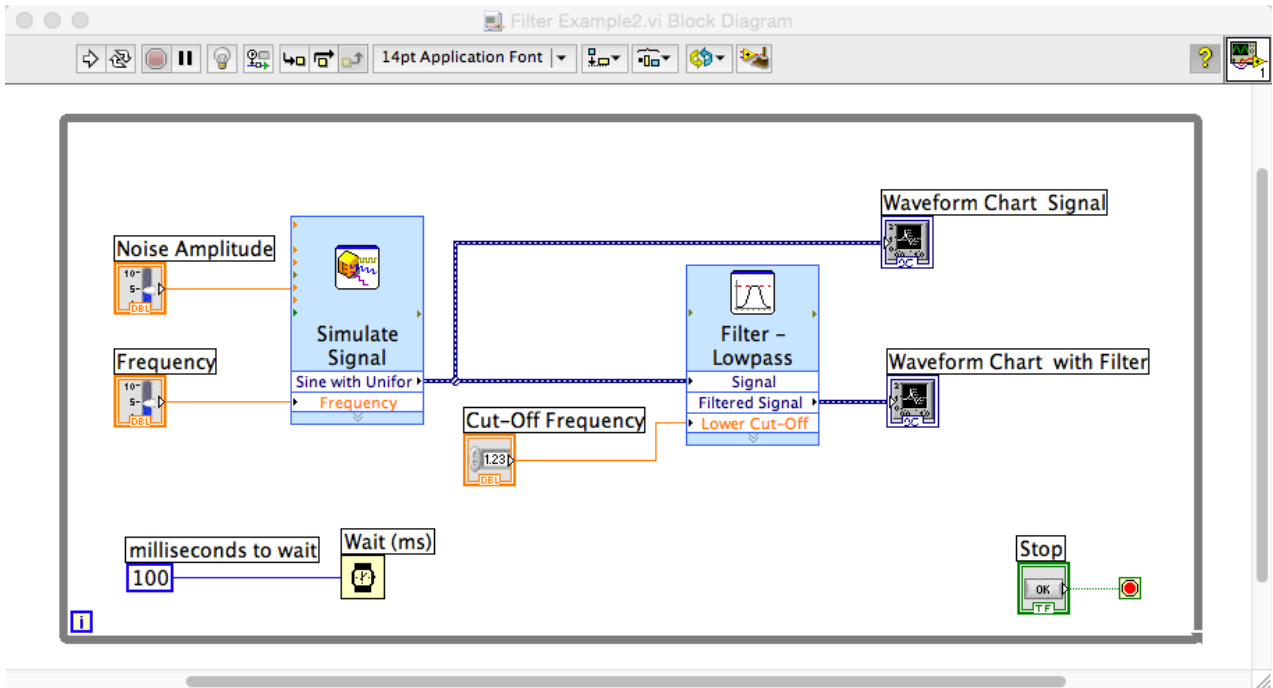


Figure 17-23: LabVIEW Example using the built-in Filter function

Figure 17-24 we see the Front Panel, where we have one chart showing the signal with noise and in the other chart we have used the Filter function in order to remove the noise.

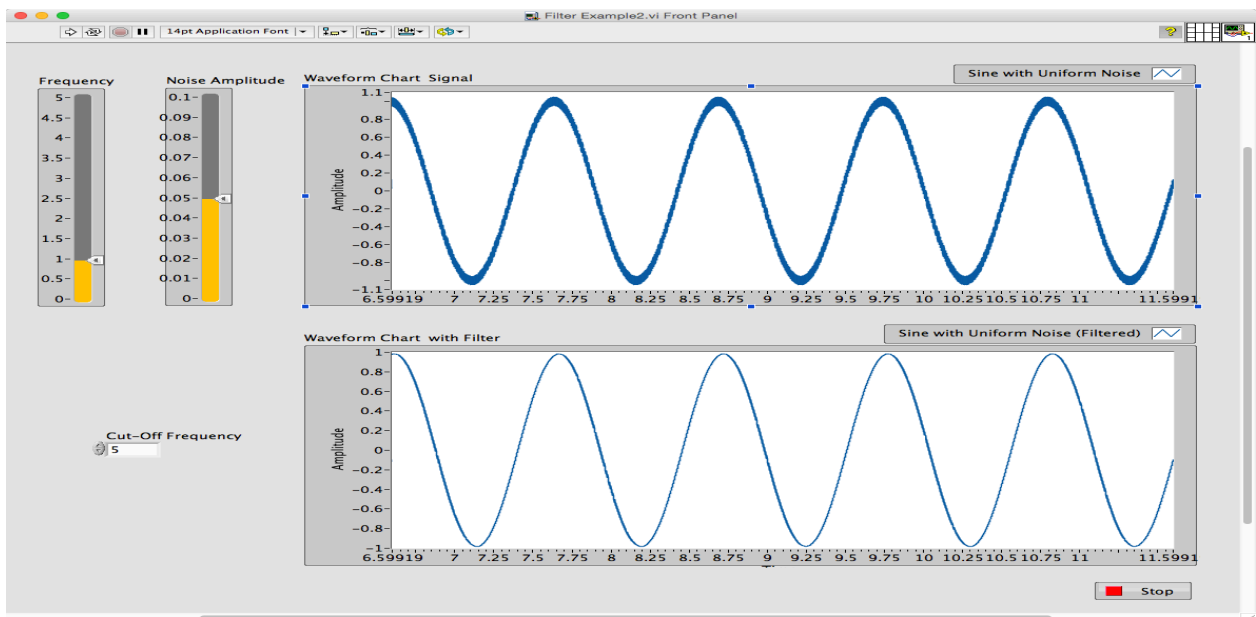


Figure 17-24: Front Panel, Example using the built-in Filter in LabVIEW

17.6.3 C# Example

17.7 Industrial DAQ Systems

A lot of Industrial DAQ systems exists today. A short introduction to some of them will be given here.

The following industrial DAQ systems are explained below.

- cDAQ (CompactDAQ)
- cRIO (CompactRIO)

Both these systems are from National Instruments.

17.7.1 cDAQ

cDAQ (CompactDAQ) (Figure 17-25) from National Instruments is a DAQ system where you can plug in different I/O modules, from 1 up to 8 depending on the hardware.



Figure 17-25: cDAQ System from National Instruments

You connect the cDAQ using a standard USB cable (or Ethernet/WiFi for some of the configurations). You can program the device in LabVIEW or Visual Studio/C# using the DAQmx driver.

For more information about cDAQ, see the following web sites:

<https://en.wikipedia.org/wiki/CompactDAQ>

<http://www.ni.com/compactdaq>

17.7.2 cRIO

cRIO (CompactRIO) (Figure 17-26) from National Instruments is similar to cDAQ, but the main difference is that you program it on your PC and then you deploy the program to the device so it can run independently from the PC, a so-called embedded system.



Figure 17-26: cRIO System from National Instruments

You need to install the “LabVIEW Real-Time Module” and the “NI RIO Driver” in order to use it within LabVIEW.

For more information about cRIO, see the following web sites:

<https://en.wikipedia.org/wiki/CompactRIO>

<http://www.ni.com/compactrio/>

A small-scale version of cRIO is called myRIO which is for more personal, non-industrial use.

<http://www.ni.com/myrio>

For more information about industrial DAQ systems and real-time systems, see the Tutorial “OPC and Real-Time Systems in LabVIEW”.

18 User Experience

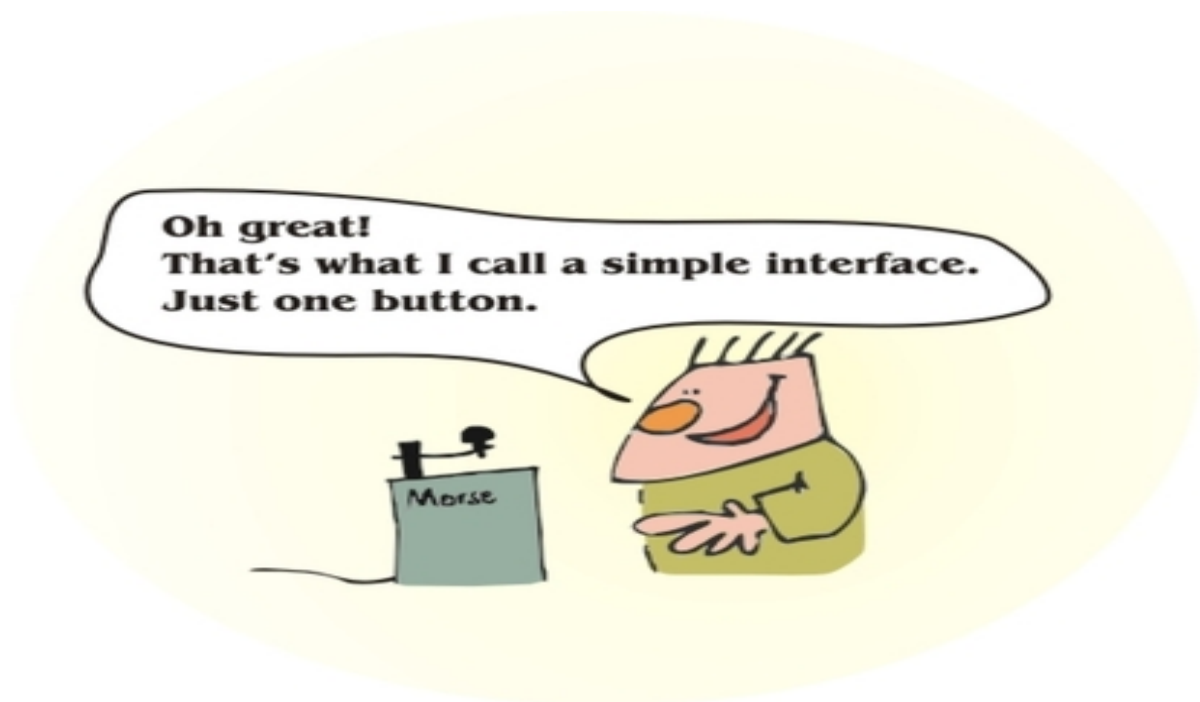
HMI (Human Machine Interface), GUI (Graphical user Interface), MMI (Man-Machine Interface), UX (User eXperience) or User Interface Design are all about creating the best experience for the user of a given system.

Some references for further reading:

https://en.wikipedia.org/wiki/User_interface

https://en.wikipedia.org/wiki/Graphical_user_interface

The GUI/HMI is the only thing the end user sees. The user does not see the code and all the other things you have created.



“Design is not just what it looks like and feels like. Design is how it works!”, Steve Jobs

“Good design is obvious. Great design is transparent”, Joe Sparano, graphic designer

“Math is easy - design is hard”, Unknown.

19 Control Systems

Figure 19-1 shows a typical control system.

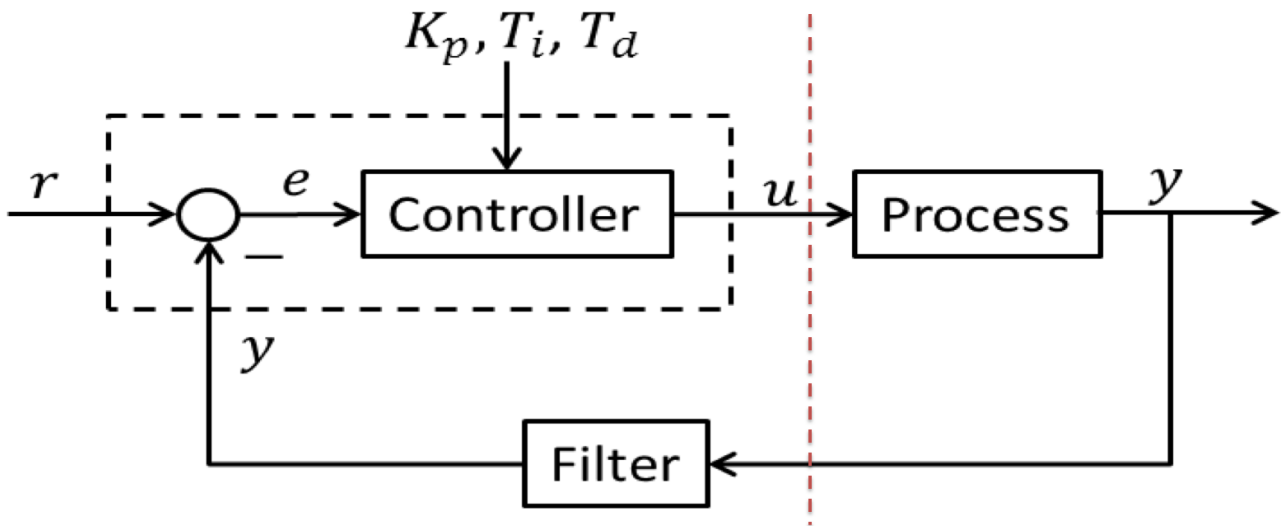


Figure 19-1: Control System

While the real process is continuous, normally the Controller and the Filter is implemented in a computer.

19.1 PC-based Control System

PC-based Control System

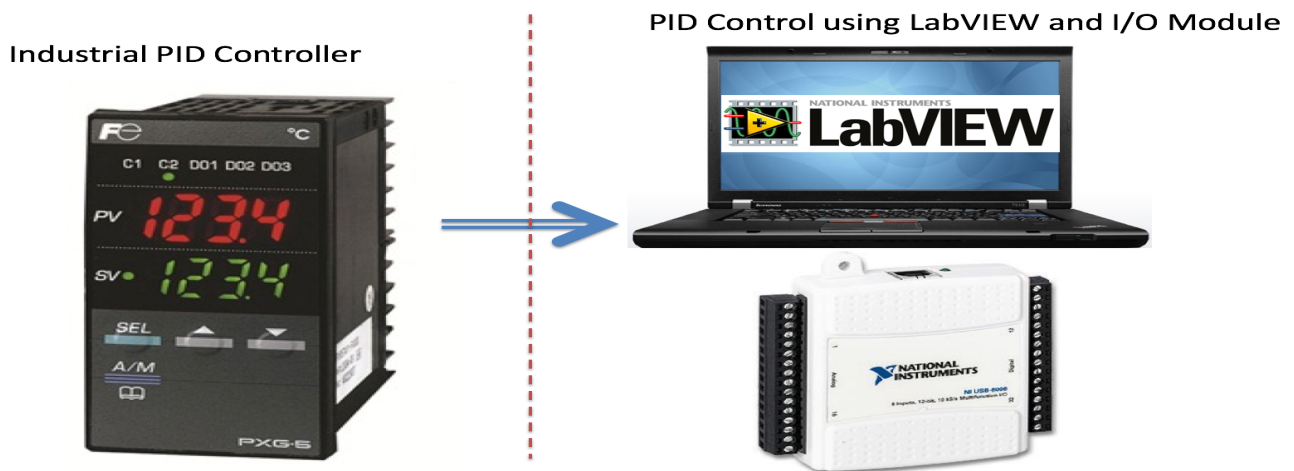


Figure 19-2: Industrial PID Controller vs. PC-based Control System

Figure 19-3 shows an example of PC-based Control System.

PC-based Control System

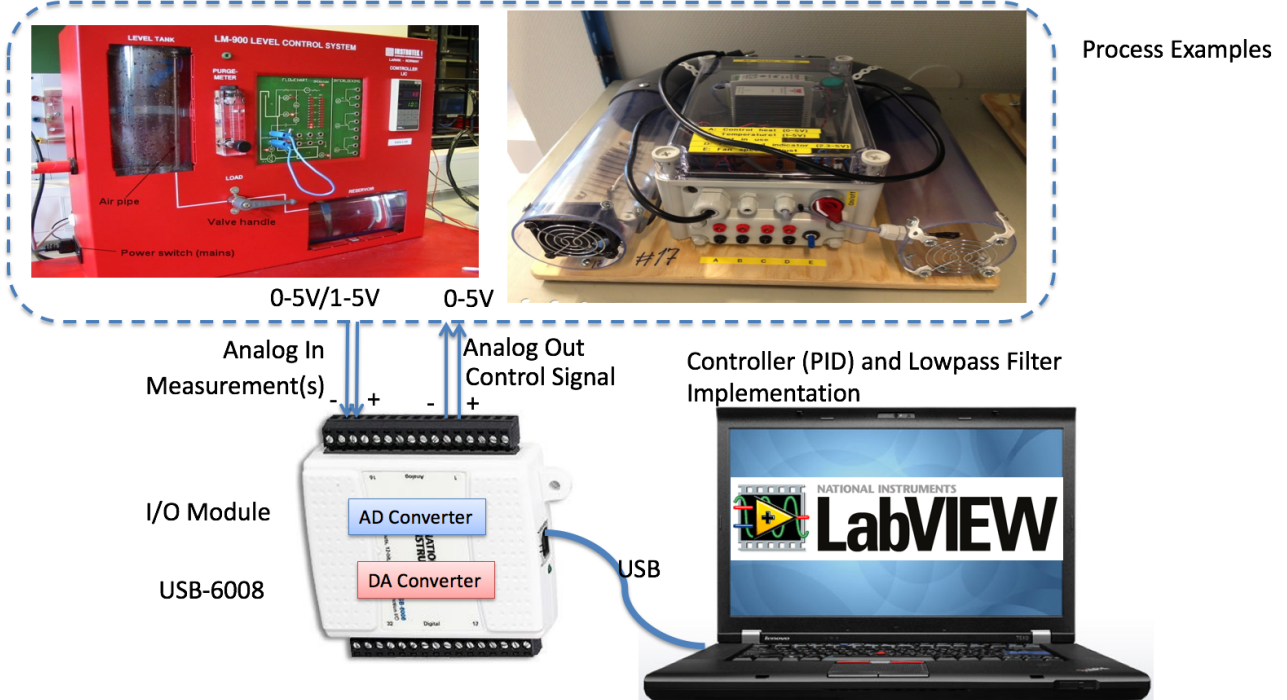
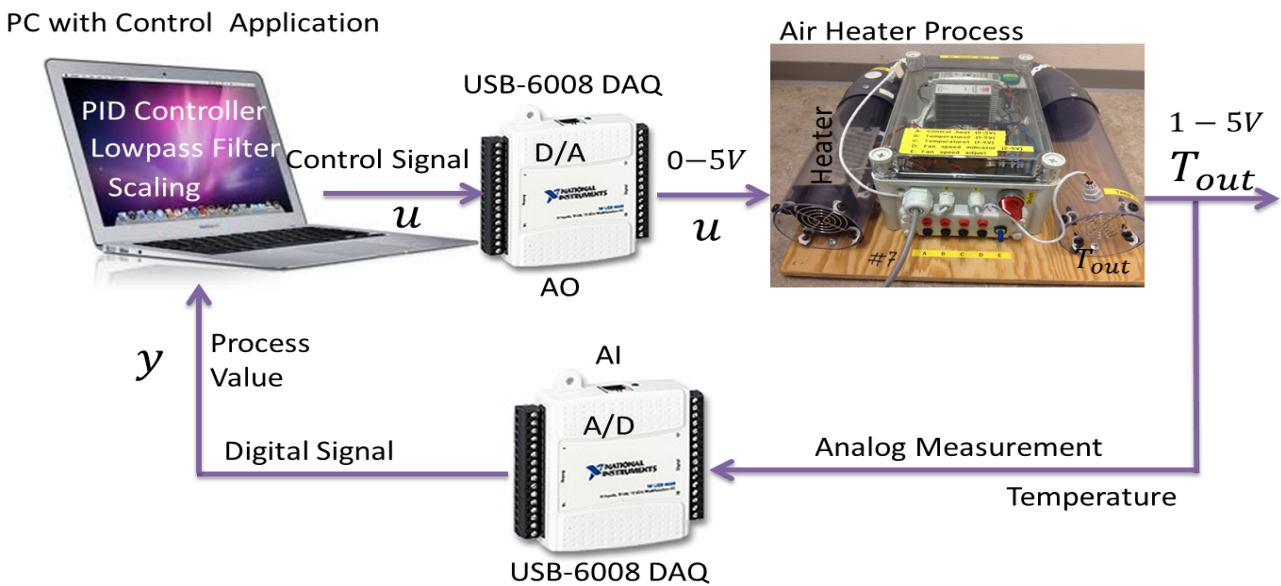


Figure 19-3: Example of PC-based Control System



19.2 PID Control

A PID controller may be written:

A PI controller may be written:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i} \int_0^t e d\tau$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

Laplace:

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

Discrete version:

We start with:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i} \int_0^t e d\tau$$

In order to make a discrete version using, e.g., Euler, we can derive both sides of the equation:

$$\dot{u} = \dot{u}_0 + K_p \dot{e} + \frac{K_p}{T_i} e$$

If we use Euler Forward we get:

$$\frac{u_k - u_{k-1}}{T_s} = \frac{u_{0,k} - u_{0,k-1}}{T_s} + K_p \frac{e_k - e_{k-1}}{T_s} + \frac{K_p}{T_i} e_k$$

Then we get:

$$u_k = u_{k-1} + u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

Where

$$e_k = r_k - y_k$$

We can also split the equation above in 2 different parts by setting:

$$\Delta u_k = u_k - u_{k-1}$$

This gives the following PI control algorithm:

$$e_k = r_k - y_k$$

$$\Delta u_k = u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

$$u_k = u_{k-1} + \Delta u_k$$

This algorithm can easily be implemented in any programming language.

19.2.1 PI Controller as a State-space model

Given:

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

We set $z = \frac{1}{s} e \Rightarrow sz = e \Rightarrow \dot{z} = e$

This gives:

$$\begin{aligned} \dot{z} &= e \\ u &= K_p e + \frac{K_p}{T_i} z \end{aligned}$$

Where

$$e = r - y$$

Discrete version:

Using Euler:

$$\dot{z} \approx \frac{z_{k+1} - z_k}{T_s}$$

Where T_s is the Sampling Time.

This gives:

$$\begin{aligned} \frac{z_{k+1} - z_k}{T_s} &= e_k \\ u_k &= K_p e_k + \frac{K_p}{T_i} z_k \end{aligned}$$

Finally:

$$\begin{aligned} e_k &= r_k - y_k \\ u_k &= K_p e_k + \frac{K_p}{T_i} z_k \\ z_{k+1} &= z_k + T_s e_k \end{aligned}$$

This algorithm can easily be implemented in any programming language.

19.3 Industrial Control Systems

Typically, we have 3 different types of Industrial Control Systems (ICS), which are (see also Figure 19-4):

- DCS – Distributed Control Systems
- PLC – Programmable Logic Controller
- SCADA – Supervisory Control and Data Logging

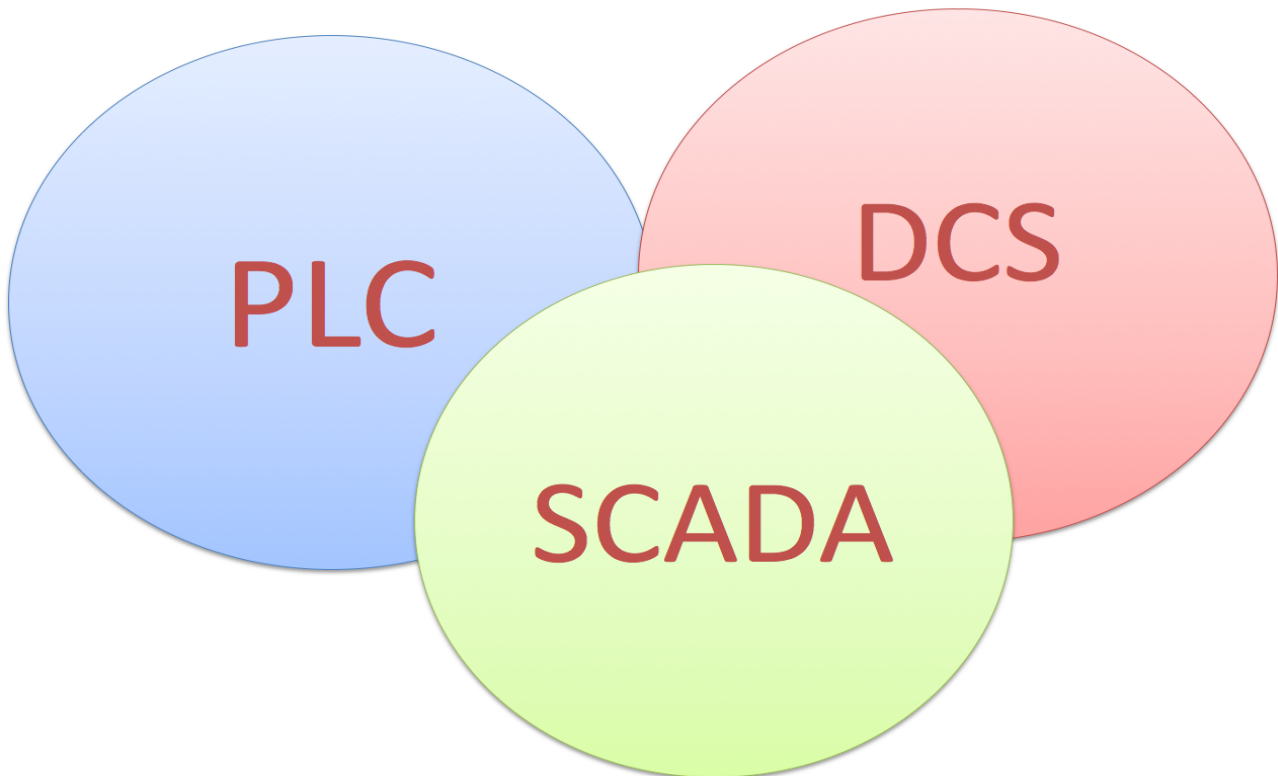


Figure 19-4: Different Types of Industrial Control Systems

Figure 19-5 shows some examples of different types of Industrial Control Systems.

Industrial Control Systems (ICS)

Industrial Control Systems are computer controlled systems that monitor and control industrial processes that exist in the physical world

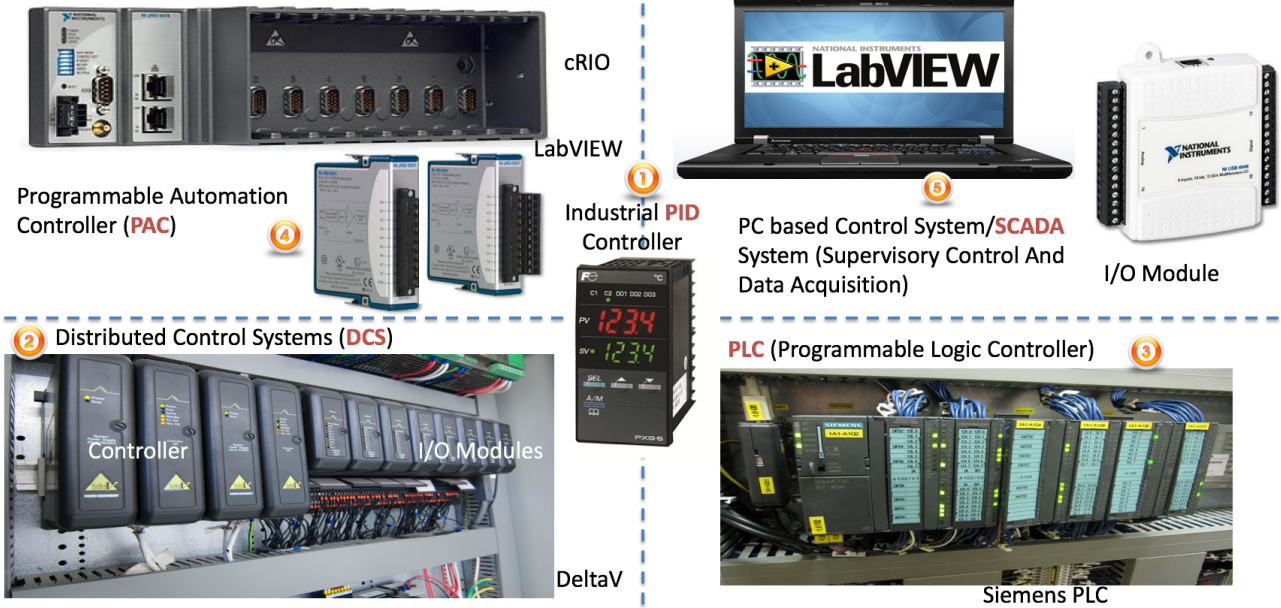


Figure 19-5: Examples of Industrial Control Systems

20 Sensors and Actuators

A **Sensor** is a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an (today mostly electronic) instrument.

<http://en.wikipedia.org/wiki/Sensor>

An **Actuator** is a type of motor for moving or controlling a mechanism or system. It is operated by a source of energy, typically electric current, hydraulic fluid pressure, or pneumatic pressure, and converts that energy into motion. An actuator is the mechanism by which a control system acts upon an environment.

<http://en.wikipedia.org/wiki/Actuator>

In a control system we need to read values from one or more sensors, then the controller calculates the control value based on the difference between the measured values and the set-point(s), then the control signal(s) are used to control the actuators, e.g. a pump or a motor.

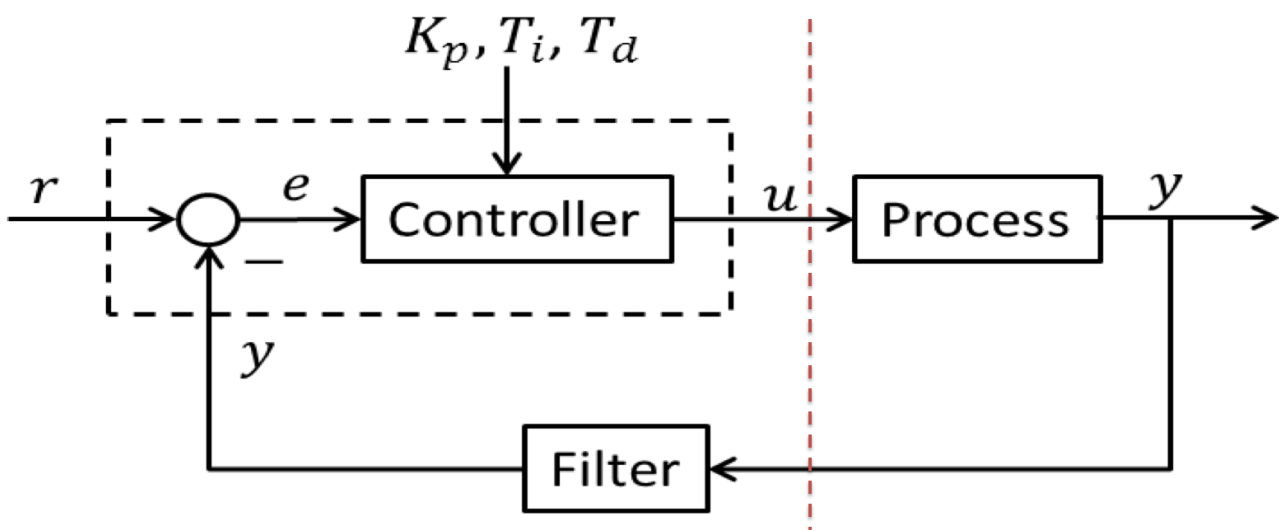


Figure 20-1: A Control System with Sensors and Actuators

20.1 Sensors

What Is a Sensor? The measurement of a physical phenomenon, such as the temperature of a room, the intensity of a light source, or the force applied to an object, begins with a sensor. A sensor, also called a transducer, converts a physical phenomenon into a measurable electrical signal. Depending on the type of sensor, its electrical output can be a voltage, current, resistance, or another electrical attribute that varies over time. Some sensors may require additional

components and circuitry to properly produce a signal that can accurately and safely be read by a DAQ device. See Figure 20-2.

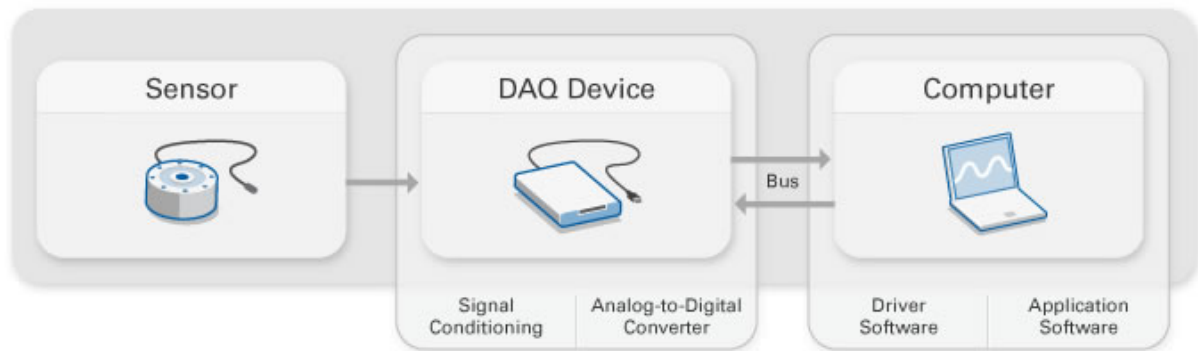


Figure 20-2: Acquire Sensor Data

Figure 20-3 shows different types of sensors.

Sensor	Phenomenon
Thermocouple, RTD, Thermistor	Temperature
Photo Sensor	Light
Microphone	Sound
Strain Gage, Piezoelectric Transducer	Force and Pressure
Potentiometer, LVDT, Optical Encoder	Position and Displacement
Accelerometer	Acceleration
pH Electrode	pH

Figure 20-3: Different Types of Sensors

For more information, please see the following links:

What is Data Acquisition? <http://www.ni.com/data-acquisition/what-is/>

Measurement Fundamentals: <http://www.ni.com/white-paper/4523/en/>

Sensor Fundamentals: <http://www.ni.com/white-paper/4045/en/>

Sensor Terminology: <http://www.ni.com/white-paper/14860/en/>

Here are some important aspects regarding sensors and measurements:

Calibration: A comparison between measurements. One of known magnitude or correctness made or set with one device and another measurement made in as similar a way as possible with a

second device. The device with the known or assigned correctness is called the standard. The second device is the unit under test, test instrument, or any of several other names for the device being calibrated.

Resolution: The smallest change it can detect in the quantity that it is measuring. The following formula may be used (where S is the measurement span, e.g., 0-100deg.C):

$$R = \frac{S}{2^n - 1}$$

Accuracy: How close the measured value is the actual/real value, e.g., ±0.1 %

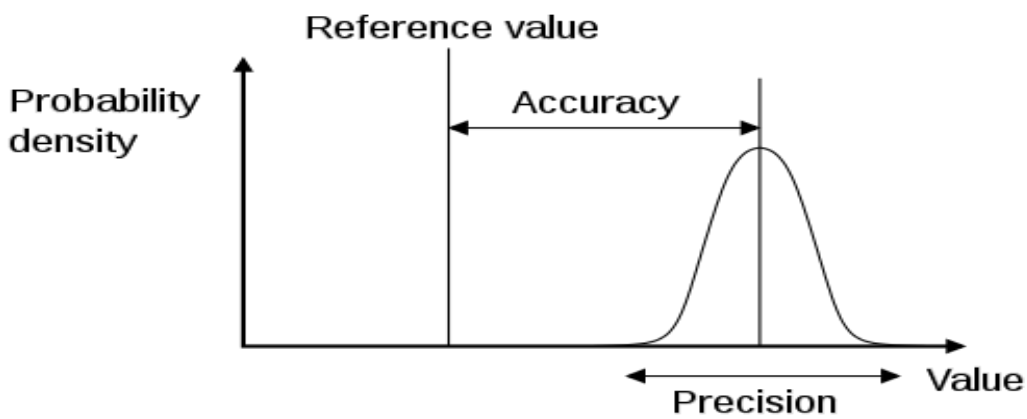


Figure 20-4: Measurement Properties

Figure 20-5 illustrates the difference between accuracy and precision.

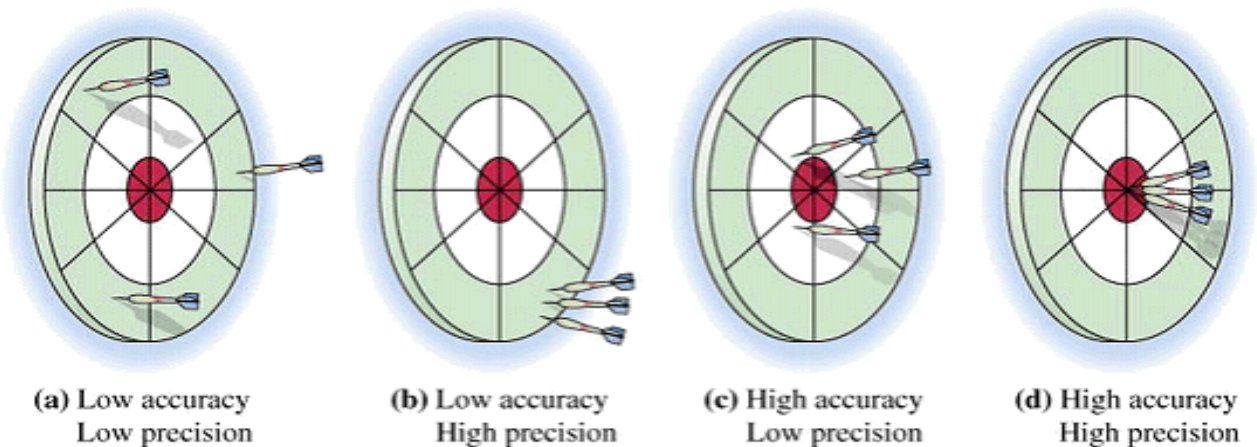


Figure 20-5: Accuracy and Precision within Measurements

20.1.1 Pt-100

Figure 20-6 shows a Pt-100 sensor with transmitter and wiring. This mounting makes it possible to connect it directly to a 0-5 DAQ device.

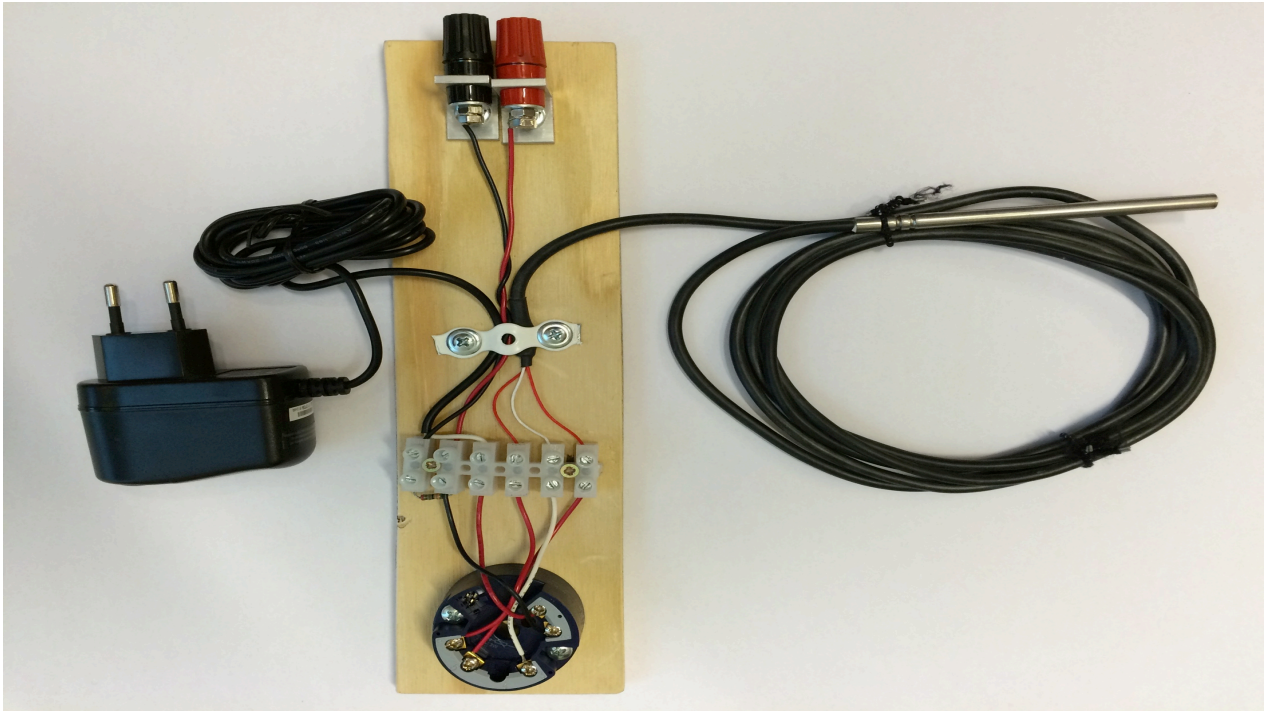


Figure 20-6: Pt-100 Measurements

Figure 20-7 shows a 3-wire Pt-100 sensor, which we use in this example.



Figure 20-7: Pt-100 with 3-wire

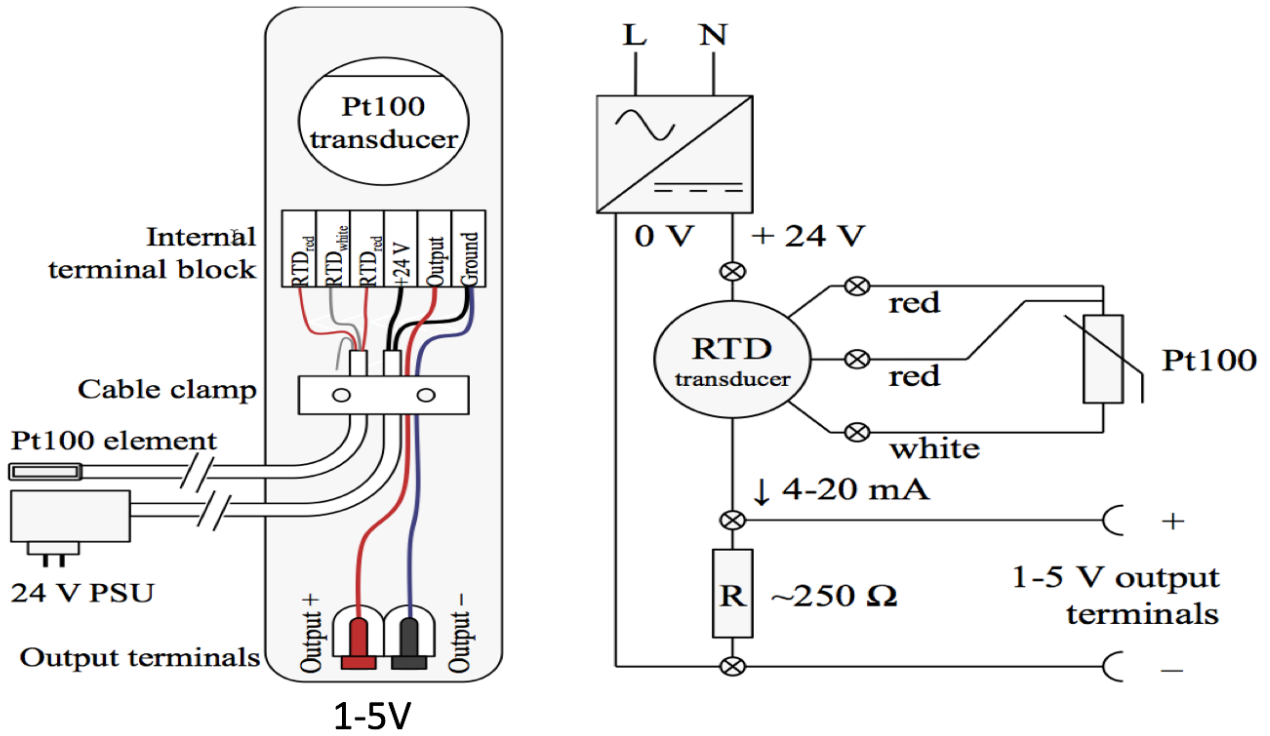


Figure 20-8: Pt-100 Wiring Schema

Figure 20-9 shows the Transmitter used in this example. The transmitter is bought from Elfa/Distrelec. A link to the transmitter is found below:

<http://www.distrelec.biz/en/temperature-signal-converter-jumo-00394779/p/17689574>



Technical data	
Operating voltage	7...30 VDC
Temperature measurement range	0...100 °C
Input, resistance thermometer	Pt100
Output, analogue	4...20 mA
Operating temperature	-40...+85 °C
Protection rating	IP 20, free mounting
Dimensions ø x H	44 x 21 mm
Accuracy	±0.1 %

Figure 20-9: Transmitter/Transducer

The transmitter (or transducer) is a temperature signal converter that outputs a 4 – 20mA signal, which is equivalent to a temperature measurement range of 0 – 100°C.

We want the output to be a voltage signal so we can use it together with a DAQ device (e.g., a NI USB-6008 DAQ device as shown in Figure 20-10) that need a input signal between 0 – 5V as input.



Figure 20-10: NI USB-6008 I/O Module

This means that we first need to convert the current signal to a voltage signal using a 250Ω resistor. This gives a voltage signal between 1-5V which we can connect to the the DAQ device.

Next we need to convert from a 1 – 5V signal into engineering units such as Celsius., i.e., 1 – 5V signal should be equivalent to 0 – 100°C. See Figure 20-11.

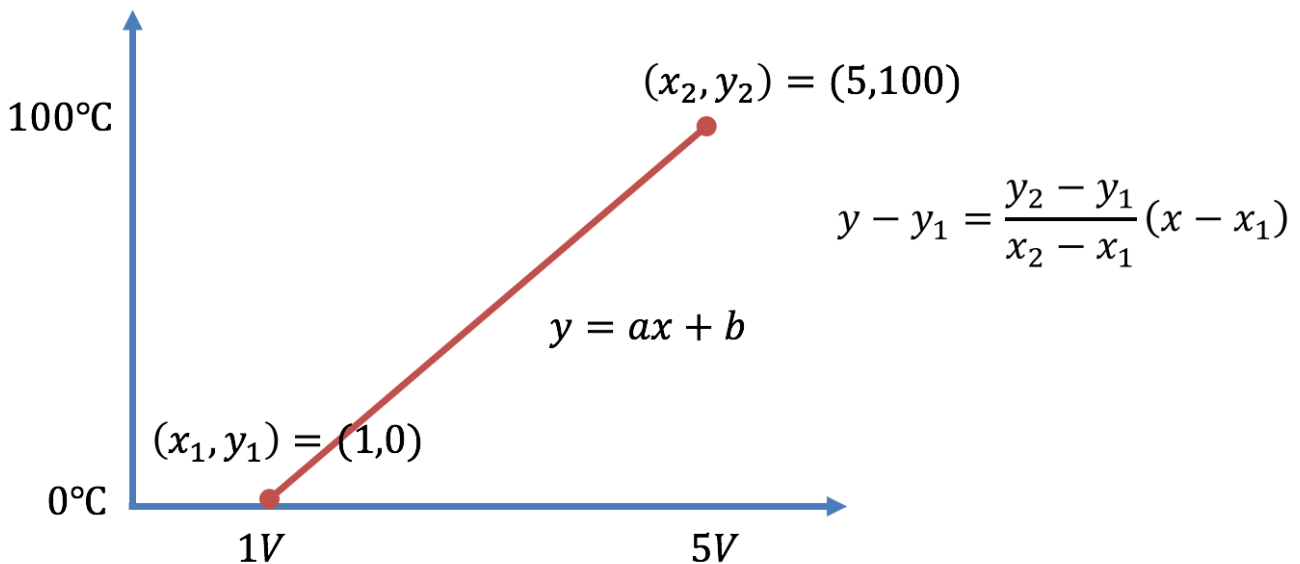


Figure 20-11: Scaling from Voltage to Engineering Units

Since we have a linear scaling:

$$y = ax + b$$

We need to find a (slope) and b (intercept):

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

We have that:

$$(x_1, y_1) = (1V, 0^\circ\text{C})$$

$$(x_2, y_2) = (5V, 100^\circ\text{C})$$

This gives:

$$y - 0 = \frac{100 - 0}{5 - 1}(x - 1)$$

Finally, we get:

$$y = 25x - 25$$

This conversion can easily be implemented in any programming language. Below we will see an example where we use LabVIEW to read values from the PT-100 device.

LabVIEW Example

We will show a LabVIEW example where we read data from this Pt-100 sensors using a NI USB-6008 DAQ device from National Instruments (see Figure 20-10).

The example includes getting data from the DAQ device, scaling from voltage to degrees Celsius and a Low-pass filter that reduces the noise.

21 OPC

Web: <https://www.halvorsen.blog/documents/technology/opc/>

21.1 What is OPC?

OPC - “Open Process Control” or “Open Platform Communications”.

A standard that defines the communication of data between devices from different manufactures

Requires an **OPC server** that communicates with the **OPC clients**

OPC allows “plug-and-play”, gives benefits as reduces installation time and the opportunity to choose products from different manufactures

Different standards: “Real-time” data (**OPC DA**), Historical data (**OPC HDA**), Alarm & Event data (**OPC AE**), etc.

In Figure 21-1 we see a typical OPC scenario.

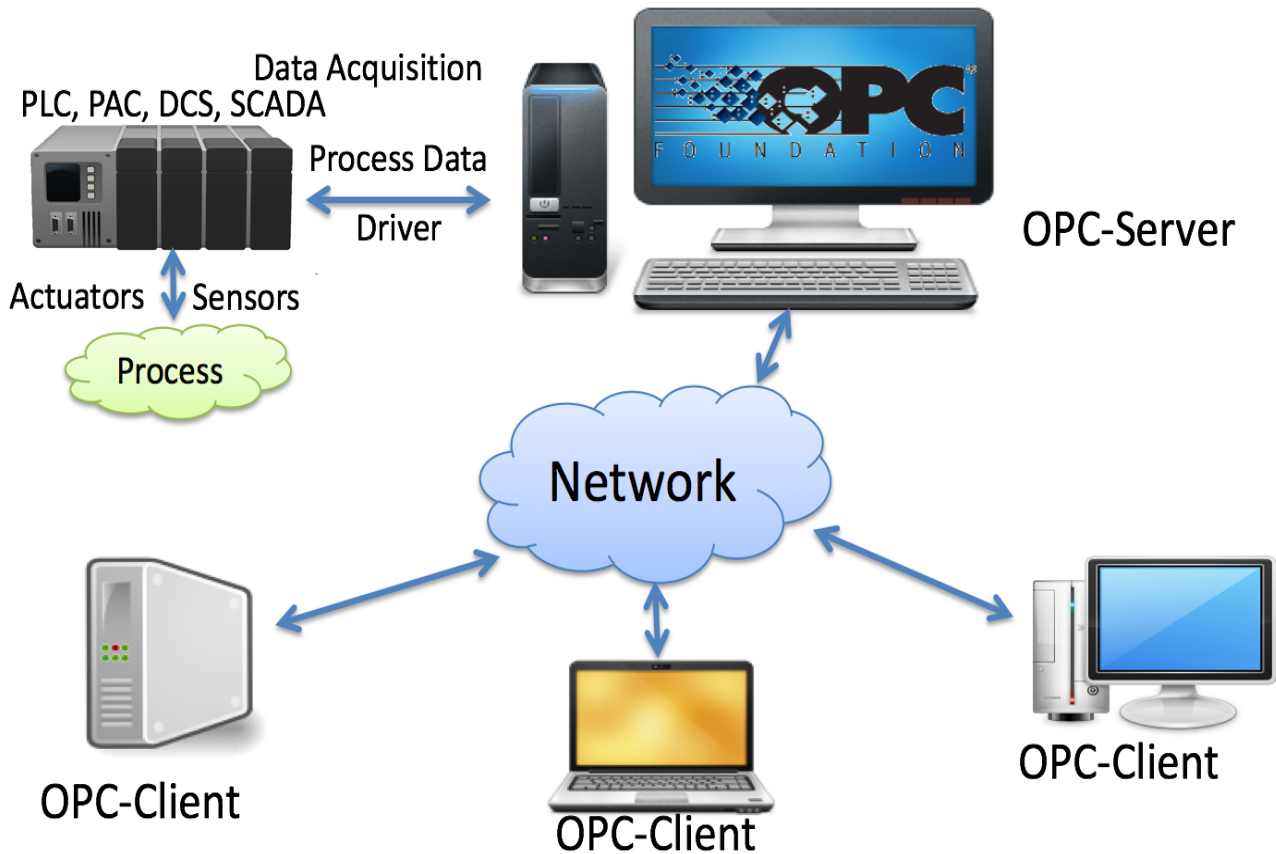


Figure 21-1: Typical OPC Scenario

21.1.1 OPC Specifications

OPC specifications:

- **OPC DA** (Data Access)

The most common OPC specification is OPC DA, which is used to read and write “real-time” data. When vendors refer to OPC generically, they typically mean OPC DA.

- OPC HDA (Historical Data Access)
- OPC A & E (Alarms & Events)
- ... (many others)

These OPC specifications are based on the OLE, COM, and DCOM technologies developed by Microsoft for the Microsoft Windows operating system family. This makes it complicated to make it work in a modern Network! Typically, you need a Tunneller Software in order to share the OPC data in a network (between OPC Servers and Clients)

- **OPC UA** (Unified Architecture)

OPC UA eliminating the need to use a Microsoft Windows based platform of earlier OPC versions. OPC UA combines the functionality of the existing OPC interfaces with new technologies such as XML and Web Services (HTTP, SOAP)

21.2 MatrikonOPC Simulation Server

A lot of OPC Servers do exist, but many of them costs a lot. The MatrikonOPC Simulation Server is free of charge. It can be used for development and test purposes not for commercial applications.

Download for free from: <http://www.matrikonopc.com>

21.2.1 MatrikonOPC Explorer (OPC Client)

Figure 21-2 shows the MatrikonOPC Explorer, which is an OPC Client for test purposes.

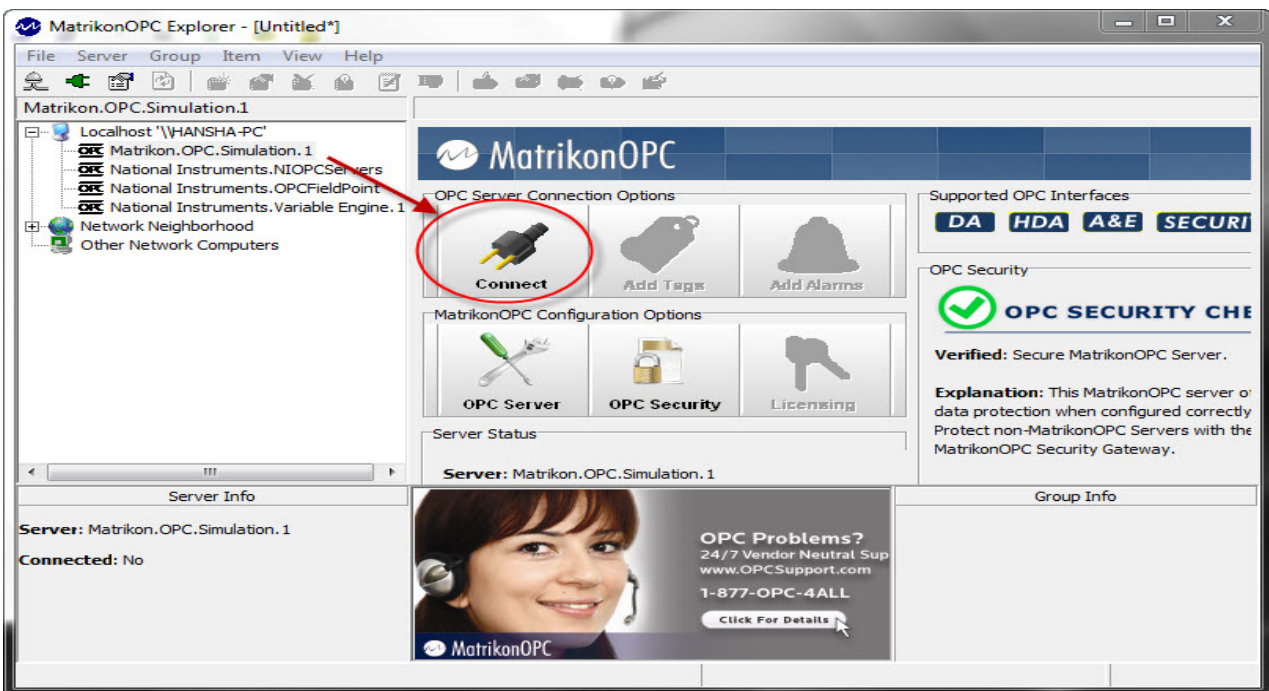


Figure 21-2: MatrikonOPC Explorer (OPC Client)

Figure 21-3 shows how to Add Tags in the MatrikonOPC Explorer.

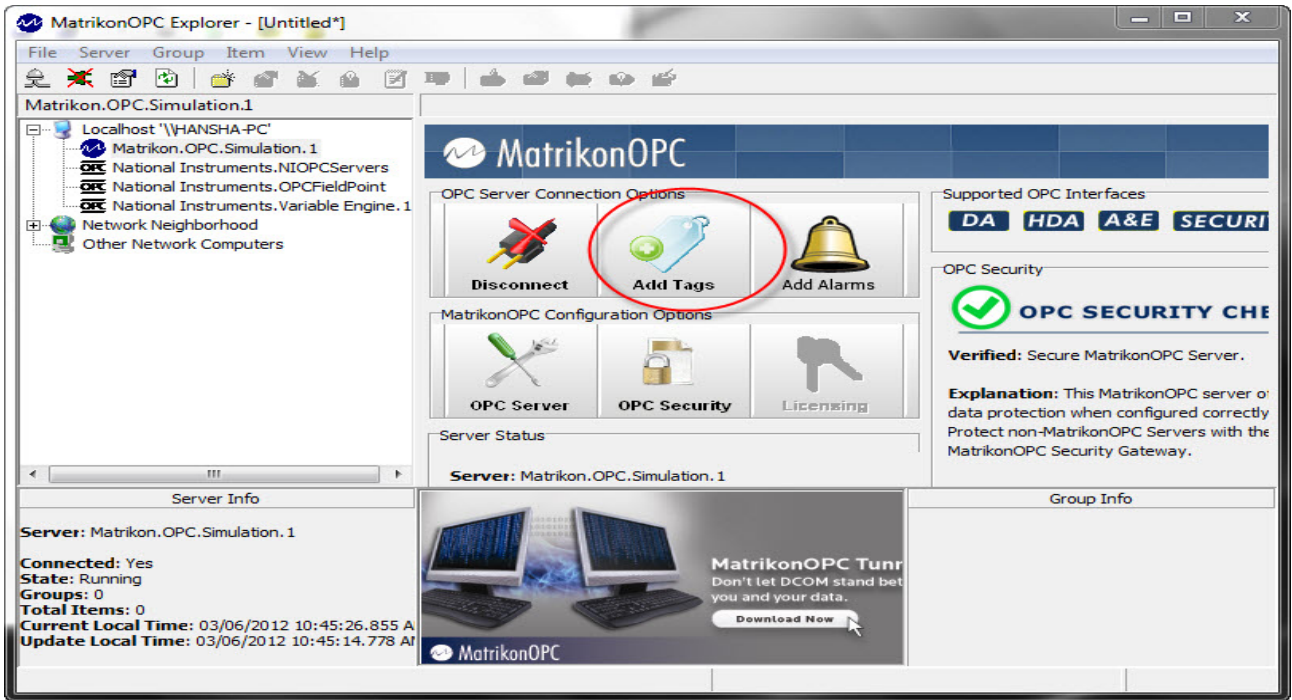
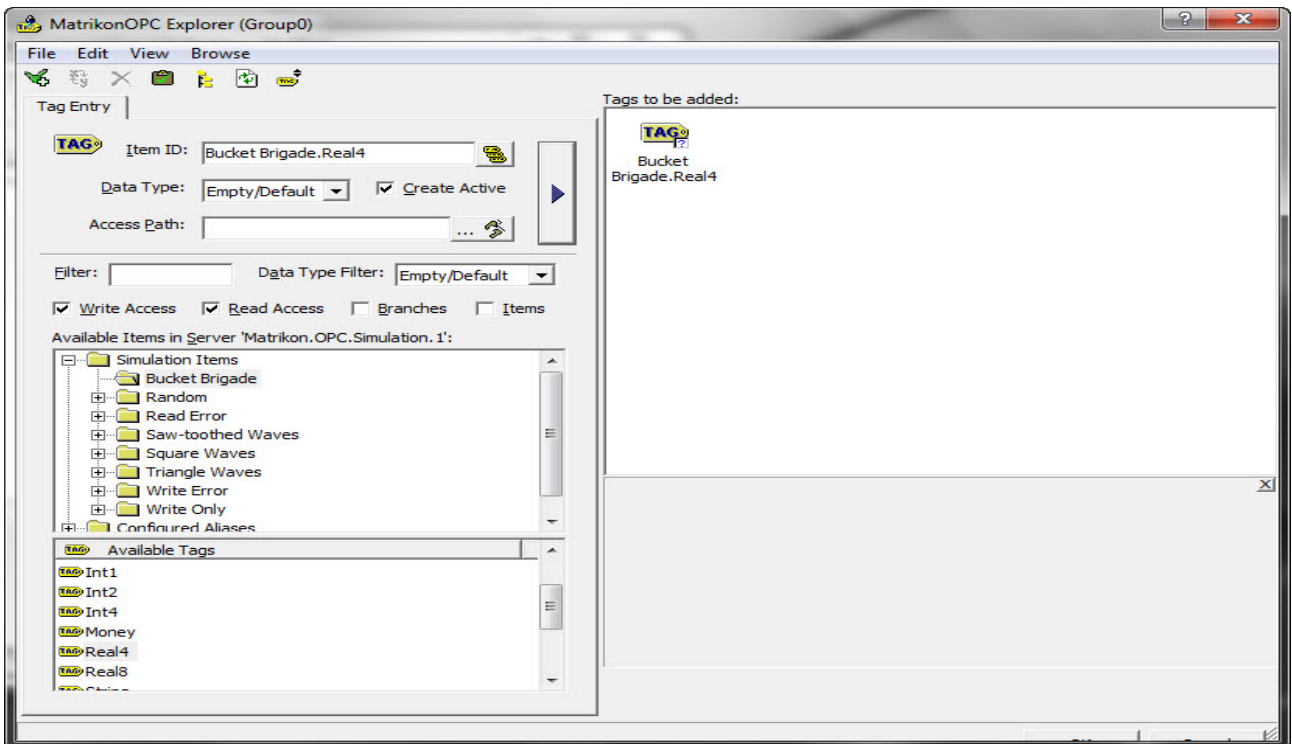


Figure 21-3: MatrikonOPC Explorer – Add Tags



21.3 OPC DA in LabVIEW

You can use LabVIEW as an OPC client by connecting to an OPC server through a **DataSocket** connection. Figure 21-4 shows the DataSocket palette in LabVIEW.

You can use LabVIEW as an OPC client by connecting to an OPC server through a **DataSocket** connection.

The **DataSocket** palette in LabVIEW:

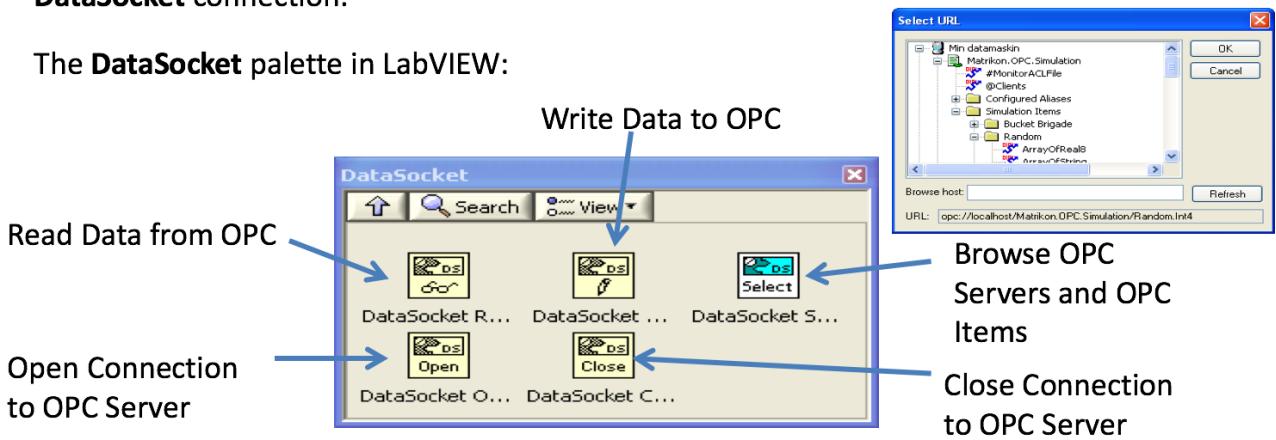


Figure 21-4: OPC DA in LabVIEW using DataSocket

In the examples below we have used the MatrikonOPC Simulation Server, but any OPC Server could have been used.

21.3.1 Write to OPC Server using LabVIEW

In Figure 21-5 we see an example of how to write to an OPC DA Server.

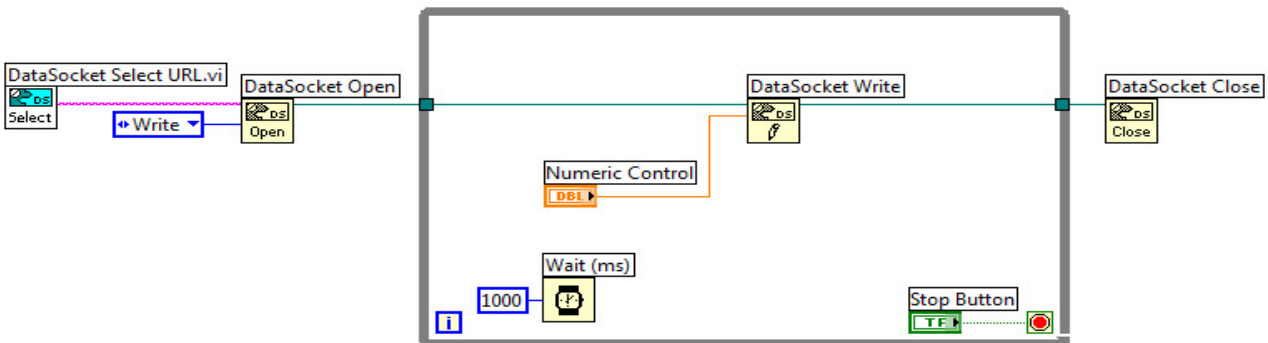


Figure 21-5: OPC DA Write in LabVIEW

Or you may specify the URL directly (Figure 21-6):

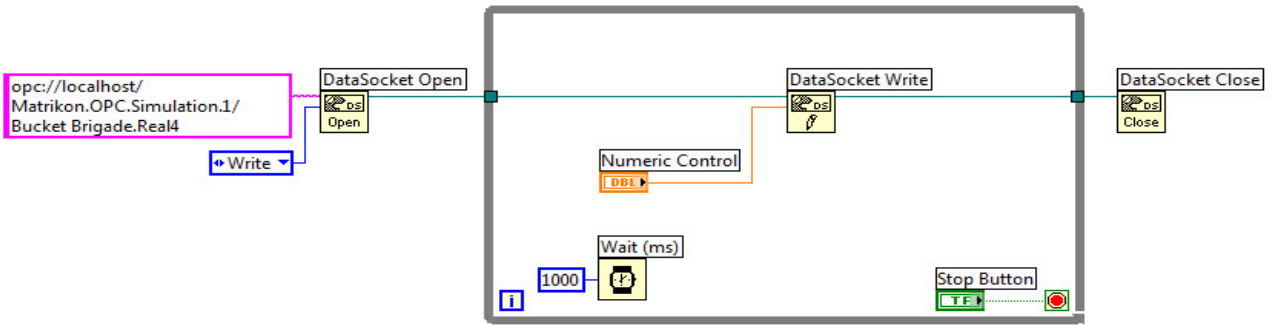


Figure 21-6: OPC DA Write in LabVIEW, Example 2

21.3.2 Read from OPC Server using LabVIEW

In Figure 21-7 we see an example of how to read from an OPC DA Server.

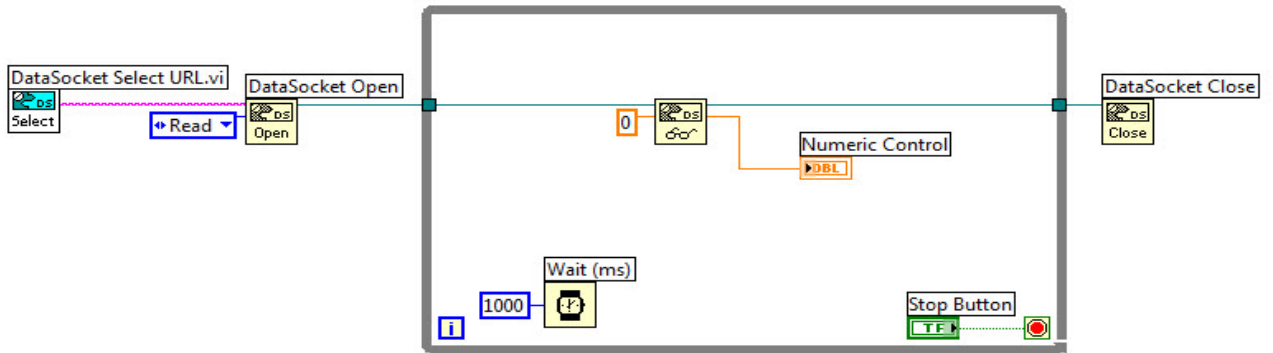


Figure 21-7: OPC DA Read in LabVIEW

Or you may specify the URL directly (Figure 21-8):

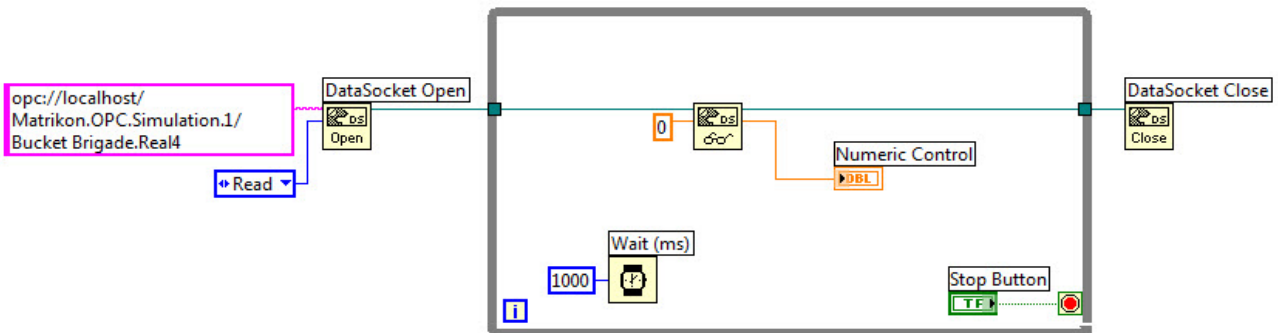


Figure 21-8: OPC DA Read in LabVIEW, Example 2

21.4 OPC DA in Visual Studio/C#

An easy way to make OPC work with Visual Studio is to install and use an add-on called “Measurement Studio”. Measurement Studio is developed by National Instruments.

Measurement Studio is an add-on to Visual Studio. Measurement Studio is used for development of measurement, control and monitoring applications using .NET and Visual Studio.

Measurement Studio has a library (DataSocket library) that makes it possible to communicate with OPC servers.

Read from OPC Server using Visual Studio:

In order to communicate with an OPC Server we can use the DataSocket API that is part of the Measurement Studio. We use the Matrikon OPC Simulation Server.

21.4.1 Read OPC Data

Below we will go through a very simple example. We will read one value from the OPC Server each time we click a button.

Visual Studio Project:

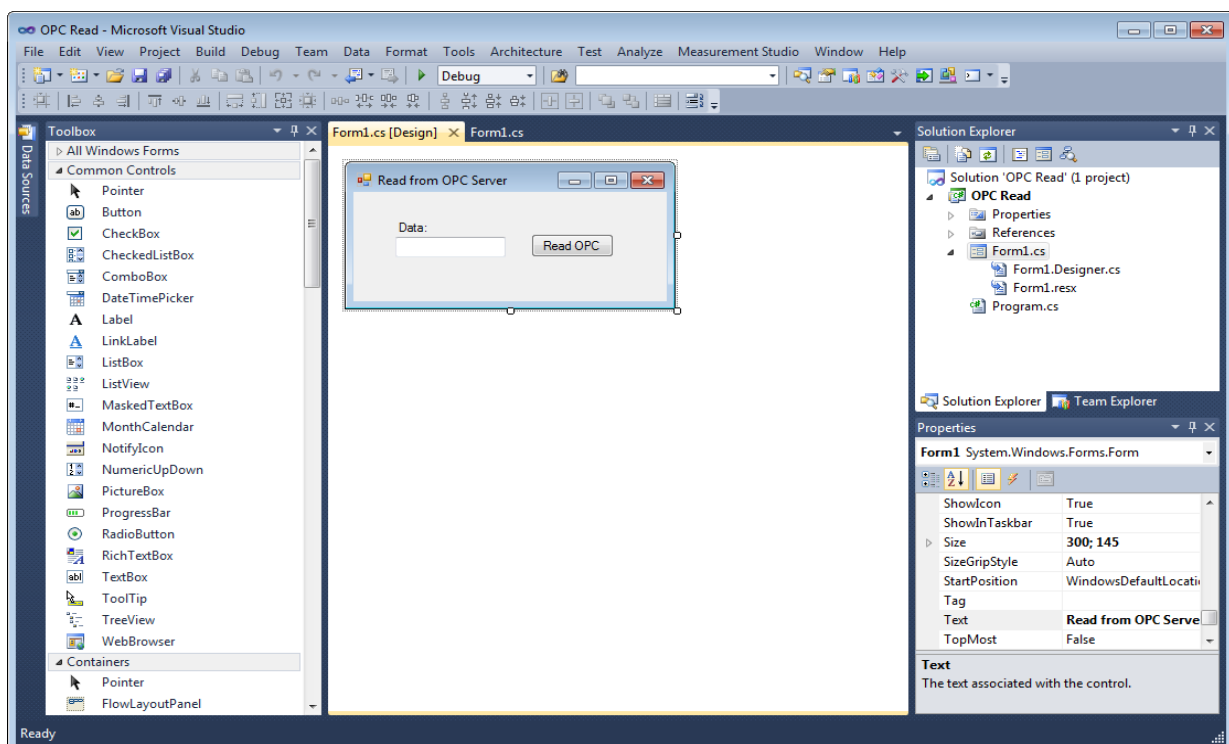


Figure 21-9: OPC Read Example in Visual Studio

Code:

We define a DataSocket object:

```
DataSocket dataSocket = new DataSocket();
```

Next, We Connect to the OPC Server:

```
string opcUrl;
opcUrl = "opc://localhost/MATRIKON.OPC.Simulation/Bucket Brigade.Real4";

if (dataSocket.IsConnected)
    dataSocket.Disconnect();

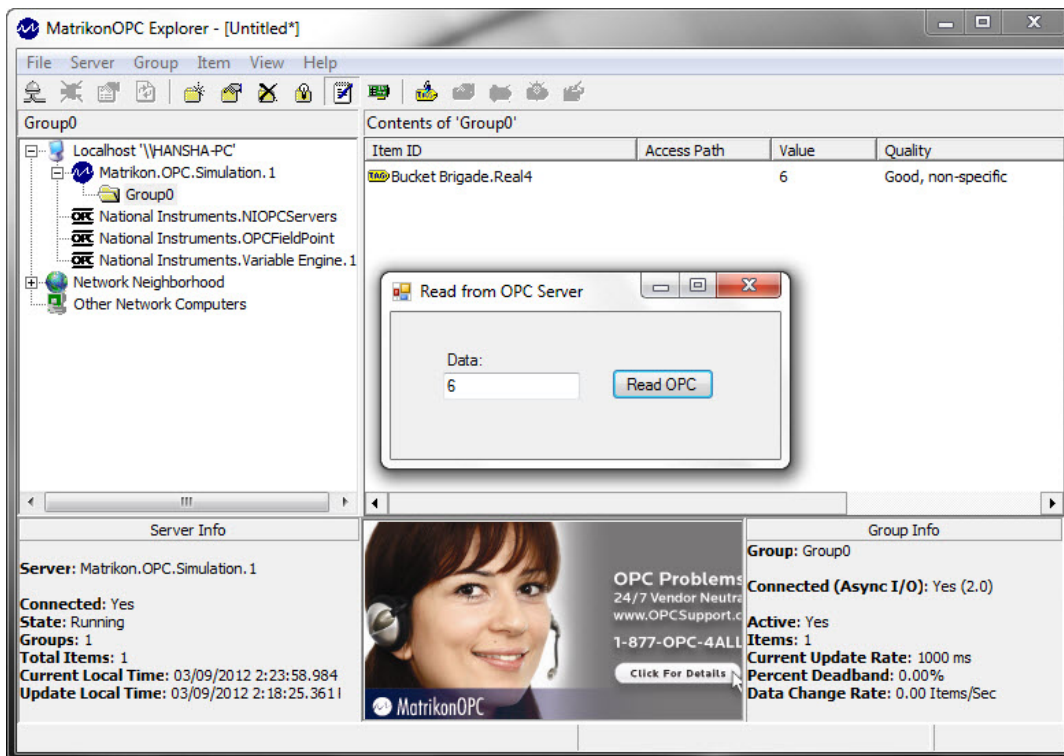
dataSocket.Connect(opcUrl, AccessMode.Read);
```

Finally, we Read OPC Data:

```
private void btnReadOpc_Click(object sender, EventArgs e)
{
    dataSocket.Update();

    txtReadOpcValue.Text = dataSocket.Data.Value.ToString();
}
```

We test the Application using the Matrikon OPC Explorer:

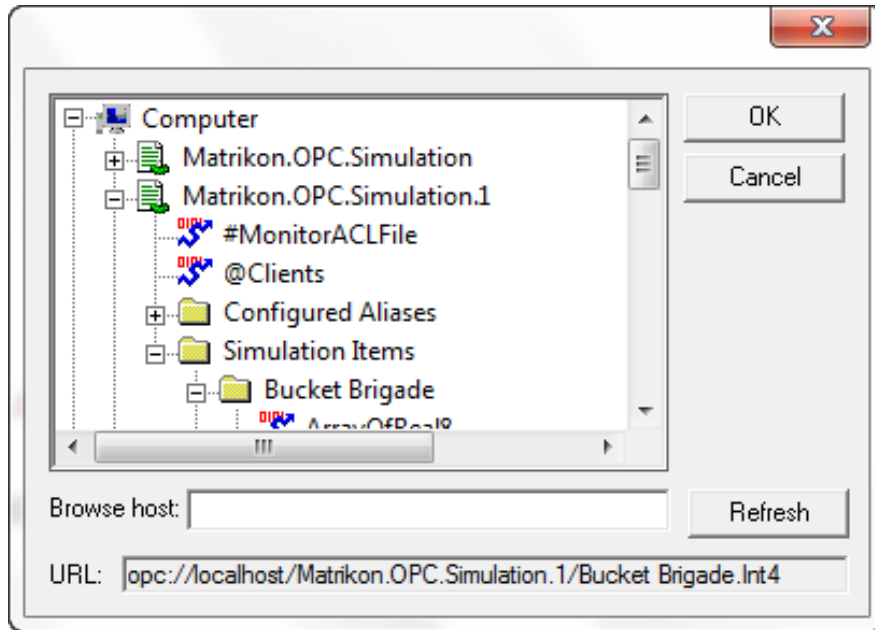


SelectUrl:

We can use the **SelectUrl** method if we want to pick the OPC item from a list of available servers (both local servers and network servers) and items.

```
dataSocket.SelectUrl();
```

The **SelectUrl** method will pop up the following window:



21.4.2 Write OPC Data

We use the same DataSocket API here.

Visual Studio Project:

Below we will go through a very simple example. We will write one value to the OPC Server each time we click a button.

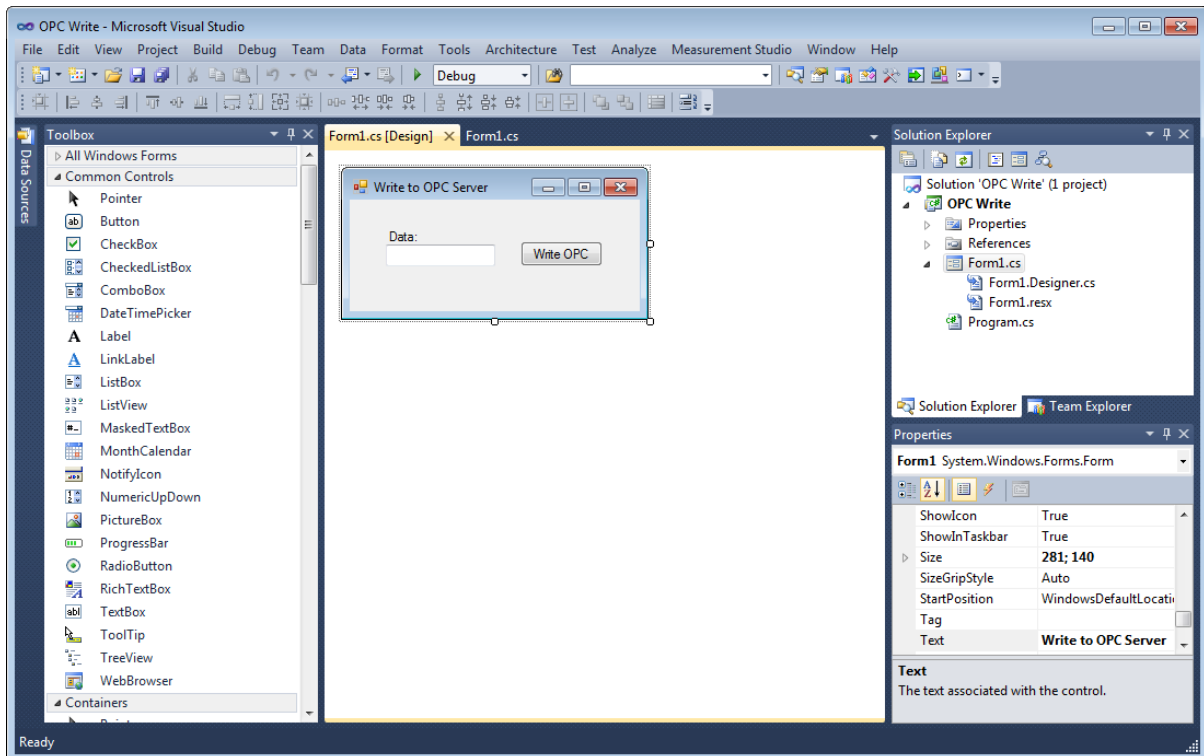


Figure 21-10: Write OPC Example in Visual studio

Code:

We define a DataSocket object:

```
DataSocket dataSocket = new DataSocket();
```

Next, We Connect to the OPC Server:

```
string opcUrl;
opcUrl = "opc://localhost/MATRIKON.OPC.Simulation/Bucket Brigade.Real4";

if (dataSocket.IsConnected)
    dataSocket.Disconnect();

dataSocket.Connect(opcUrl, AccessMode.Write);
```

Finally, we Write OPC Data:

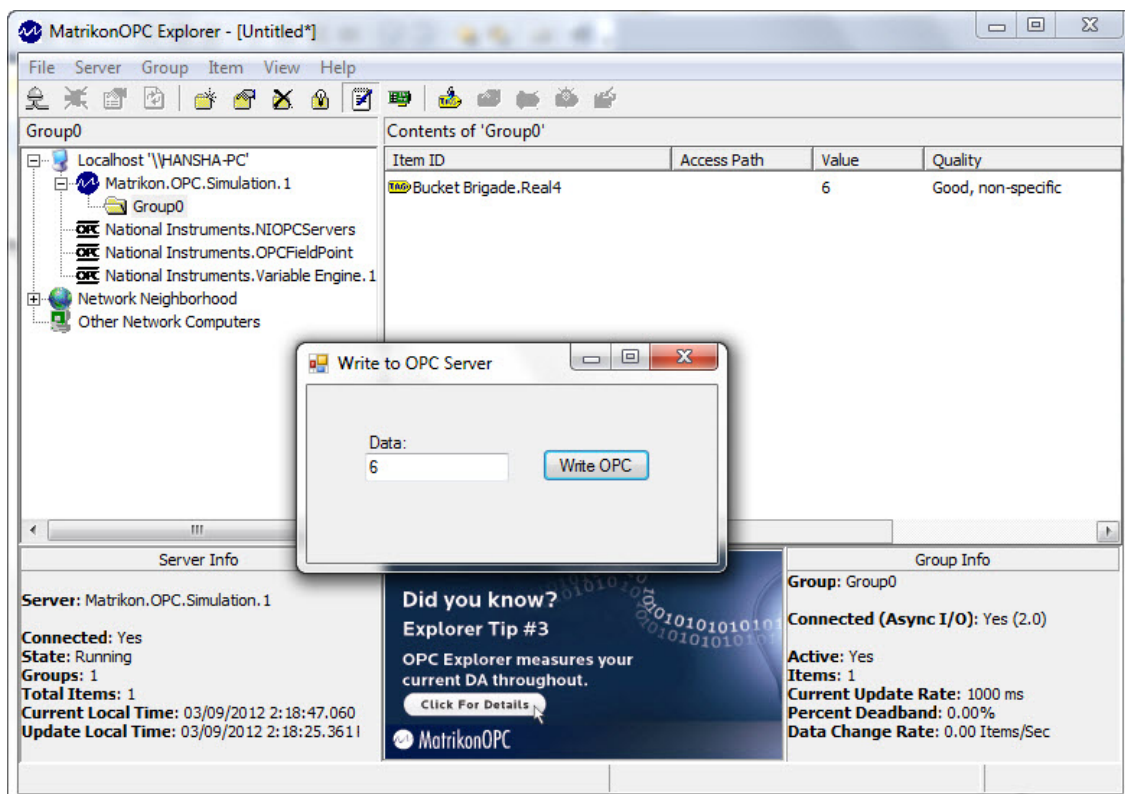
```
private void btnWriteOpc_Click(object sender, EventArgs e)
{
    double opcValue = 0;

    opcValue = Convert.ToDouble(txtWriteOpcValue.Text);

    dataSocket.Data.Value = opcValue;

    dataSocket.Update();
}
```

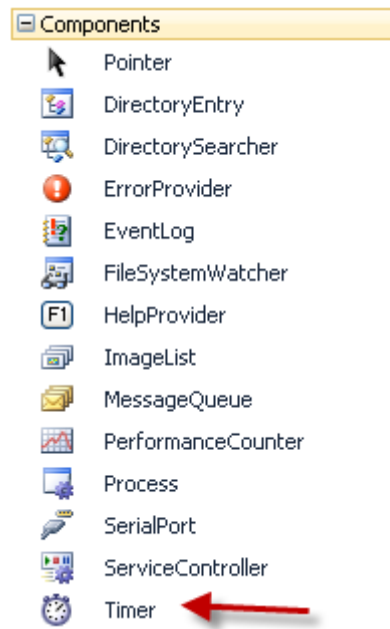
We test the Application using the Matrikon OPC Explorer:



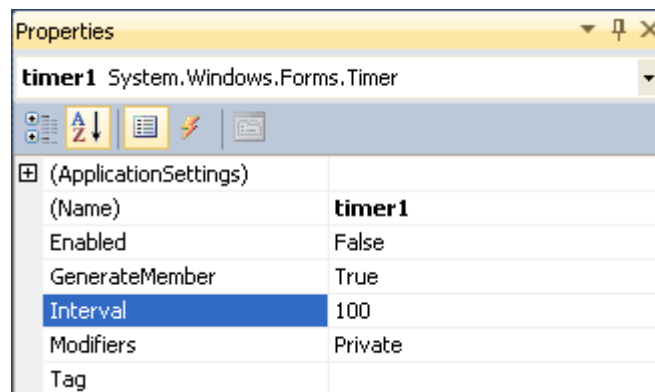
21.4.3 Using a Timer

We can use a timer in order to read values “continuously”, i.e. at specific intervals.

In the “Components” toolbox we find the “Timer” Control:



In the Properties window we can specify the Interval (“Sampling Time”) in milliseconds.



We can start the timer with the following code:

```
public Form1 ()
{
    InitializeComponent();

    timer1.Start();
}
```

In the Timer Event we create the code in order to read data at this specific interval.

```
private void timer1_Tick(object sender, EventArgs e)
```



```
{
  ...
  ...
}
```

21.5 OPC DA in MATLAB

For more information about the MATLAB OPC Toolbox: <http://se.mathworks.com/products/opc>

Acquire Data from an OPC DA Server (with Examples):

<http://se.mathworks.com/help/opc/examples/acquire-data-from-an-opc-data-access-server.html?prodcode=OT&language=en>

21.6 OPC UA

OPC UA (Unified Architecture) is the “next generation” OPC. OPC UA solves problems with standard/classic OPC (Figure 21-11).

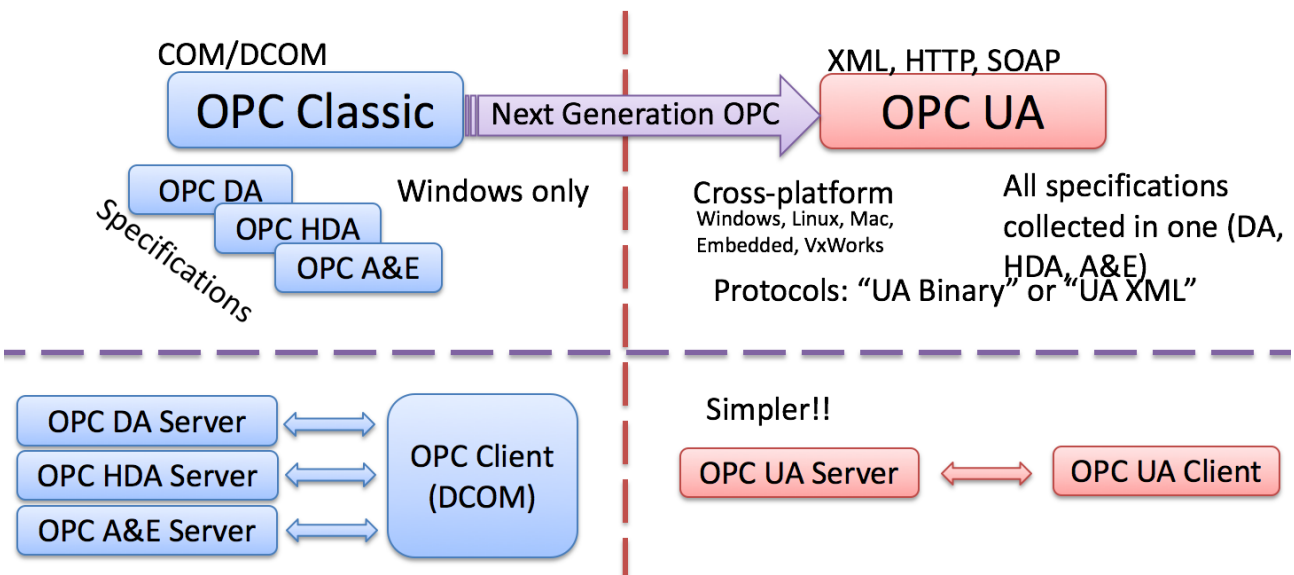


Figure 21-11: OPC DA vs. OPC UA

OPC DA limitations and challenges:

- Works only on Windows
- Cumbersome to use OPC in a network due to COM/DCOM

OPC UA has the following advantages:

- OPC UA eliminating the need to use a Microsoft Windows based platform of earlier OPC versions.
- OPC UA combines the functionality of the existing OPC interfaces with new technologies such as XML and Web Services (HTTP, SOAP)
- Cross-platform
- No dedicated OPC Server is no longer necessary because the server can run on an embedded system

Security:

OPC UA supports two protocols.

- **“UA Binary”** protocol `opc.tcp://Server`
This uses a simple binary protocol
- **“UA XML”** protocol `http://Server`
This used open standards like XML, SOAP (-> Web Service)

This is visible to application programmers only via changes to the URL. Otherwise OPC UA works completely transparent to the API.

To open DCOM (which OPC DA uses) through firewalls demanded a large hole in the firewall, which is impossible to route over the Internet!

OPC UA requires no hole in firewall (UA XML), with OPC UA Binary just a simple “needle stick” is necessary (Figure 21-12).

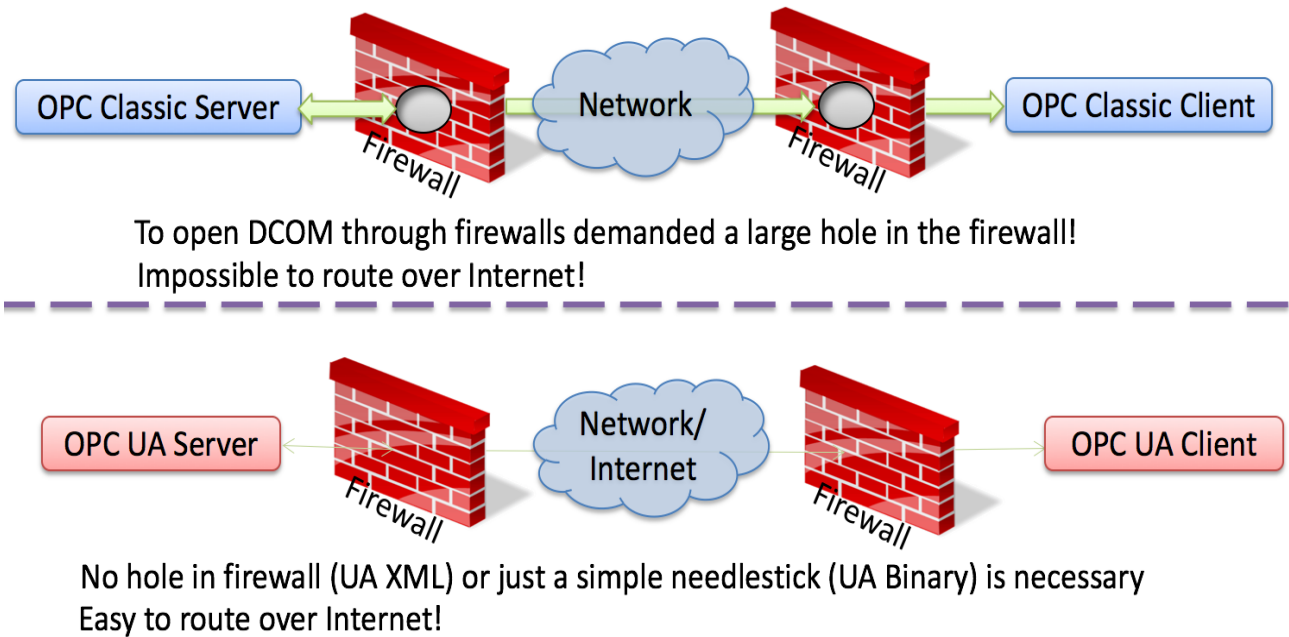


Figure 21-12: OPC UA and Firewall

OPC UA Server:

With OPC DA, the server can only run on a Windows computer, while OPC UA can run on different systems, such as embedded systems, Linux and Windows (Figure 21-13).

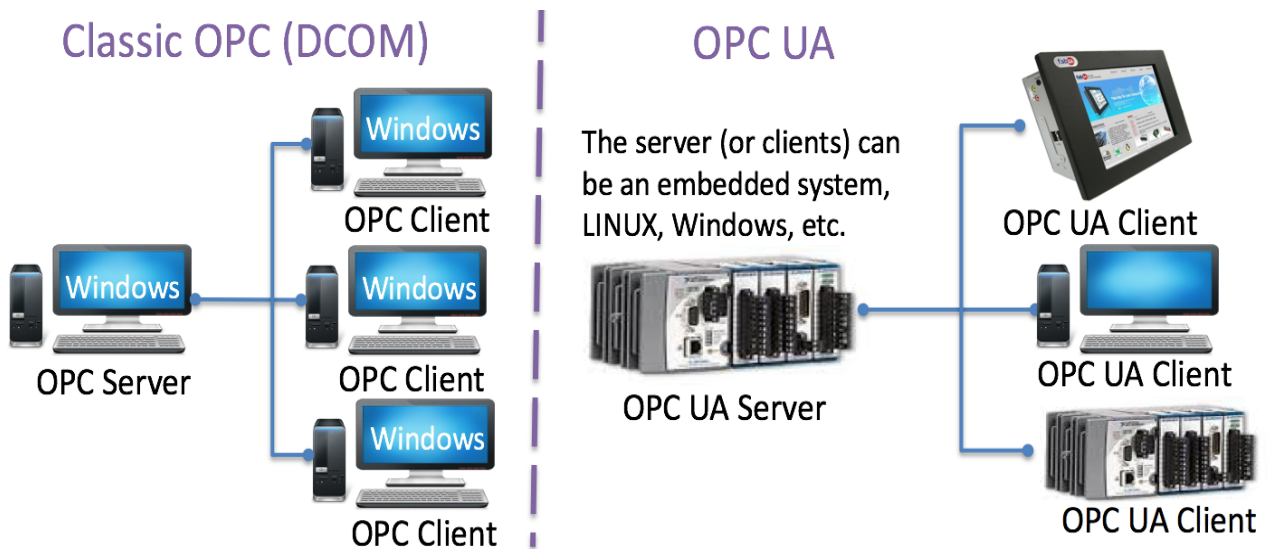


Figure 21-13: An OPC UA Server can run on Embedded Linux Systems

21.6.1 OPC UA in LabVIEW

Figure 21-14 shows the OPC UA palette in LabVIEW.

OPC UA in LabVIEW

<http://zone.ni.com/reference/en-XX/help/371618J-01/TOC9.htm>

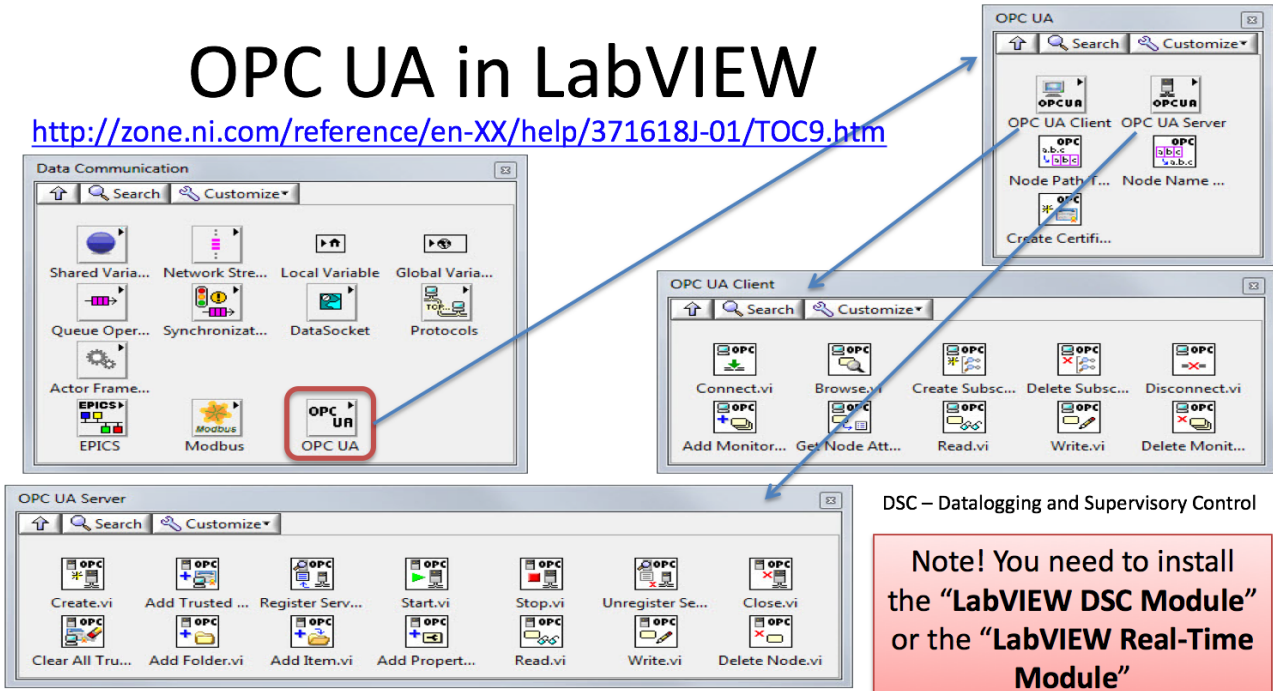


Figure 21-14: The OPC UA palette in LabVIEW

Note! In order to use the OPC UA Vis in LabVIEW you need to install the “LabVIEW DSC Module” or the “LabVIEW Real-Time Module”.

Using the LabVIEW OPC UA palette we can create both OPC UA Servers and OPC UA clients.

OPC UA Server

LabVIEW OPC UA Server Example (see Figure 21-15):

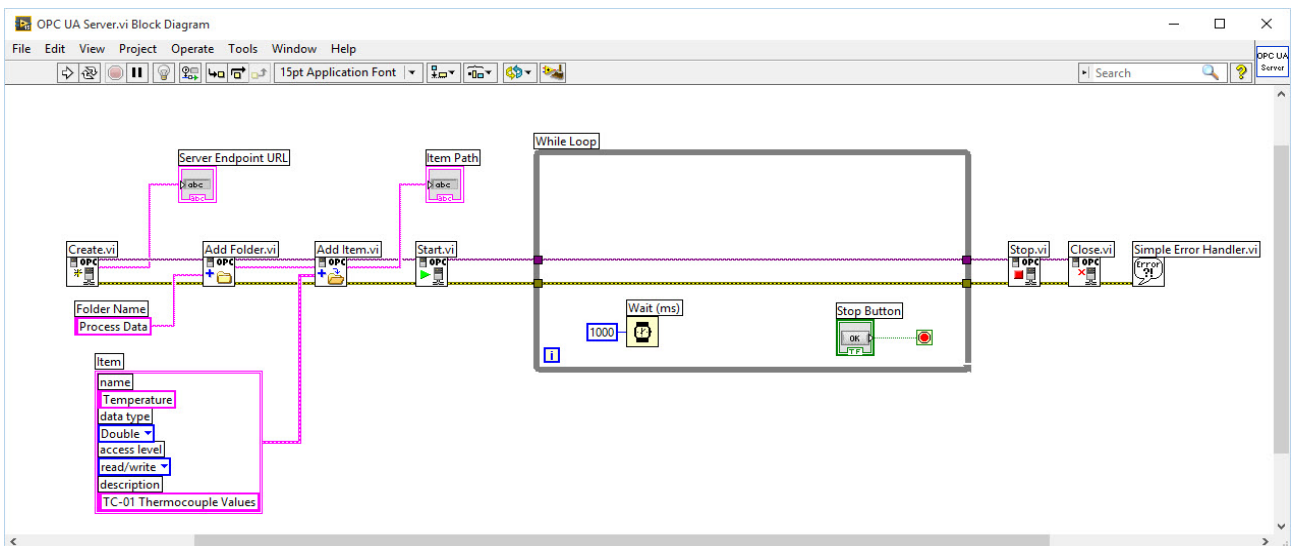


Figure 21-15: OPC UA Server Example – Block Diagram

OPC UA Clients

LabVIEW OPC UA Client Write Example (Figure 21-16):

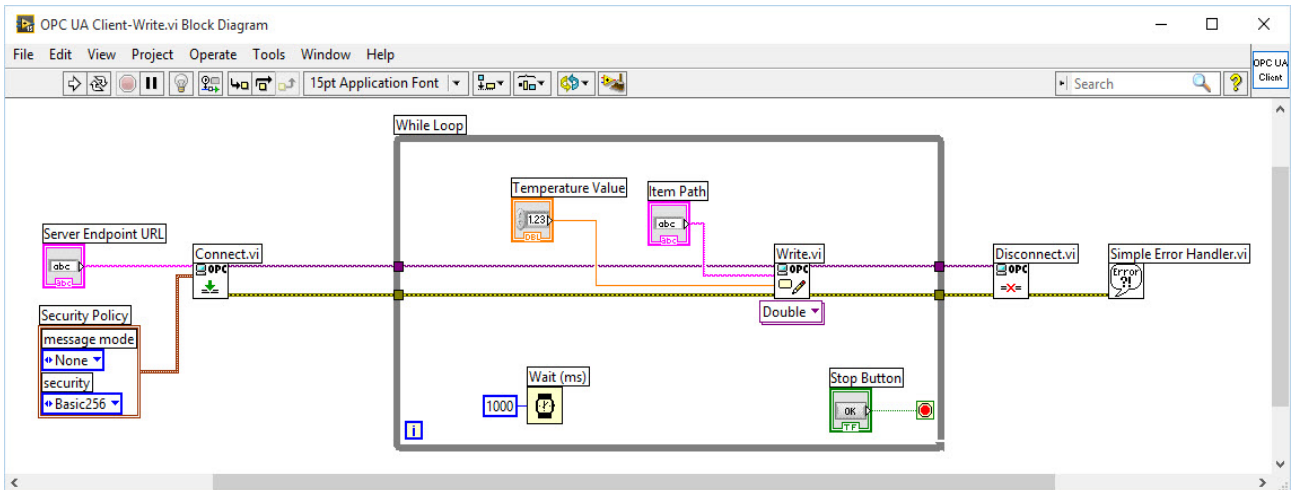


Figure 21-16: OPC UA Client – Write to OPC UA Server

LabVIEW OPC UA Client Read Example (Figure 21-17):

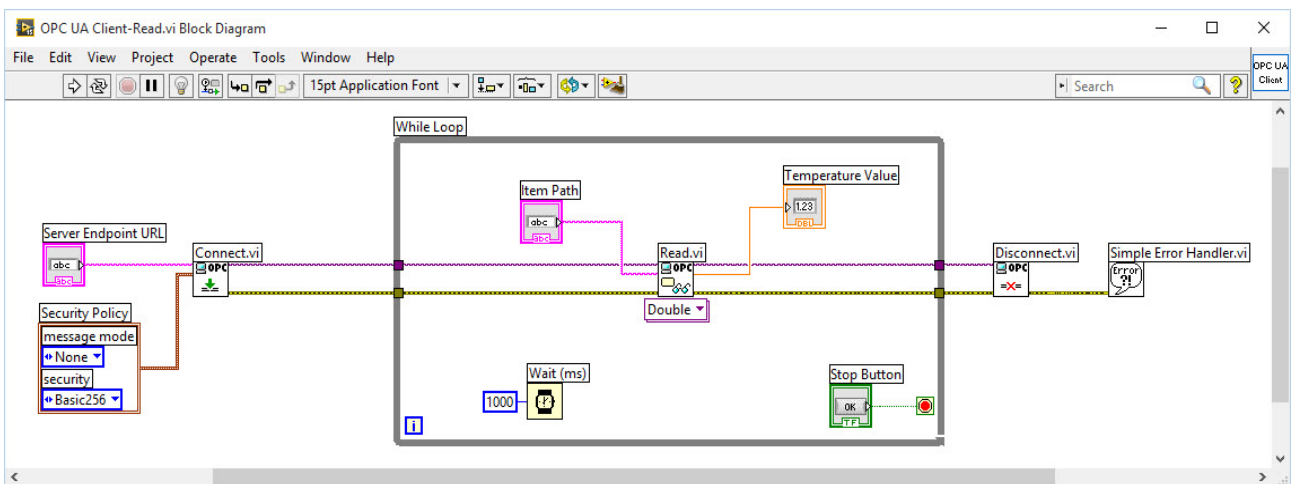


Figure 21-17: OPC UA Client – Read from OPC UA Server

For test purpose we start by running all these 3 programs at the same time on the same computer. If everything works as expected, we can then try to install them on 3 different computers in a network.

21.6.2 OPC UA in MATLAB

Until MATLAB R2015a only OPC DA was supported. From MATLAB R2015b also OPC UA is supported.

22 SCADA Systems

Web: <https://www.halvorsen.blog/documents/technology/scada/>

22.1 Introduction

SCADA (Supervisory Control And Data Acquisition) is a type of Industrial Control System (ICS).

Industrial Control Systems (ICS) are computer controlled systems that monitor and control industrial processes that exist in the physical world.

We typically can divide Industrial Control Systems into 3 categories:

- PLC systems
- DCS systems
- SCADA systems

Figure 20-1 shows an overview of the different types.

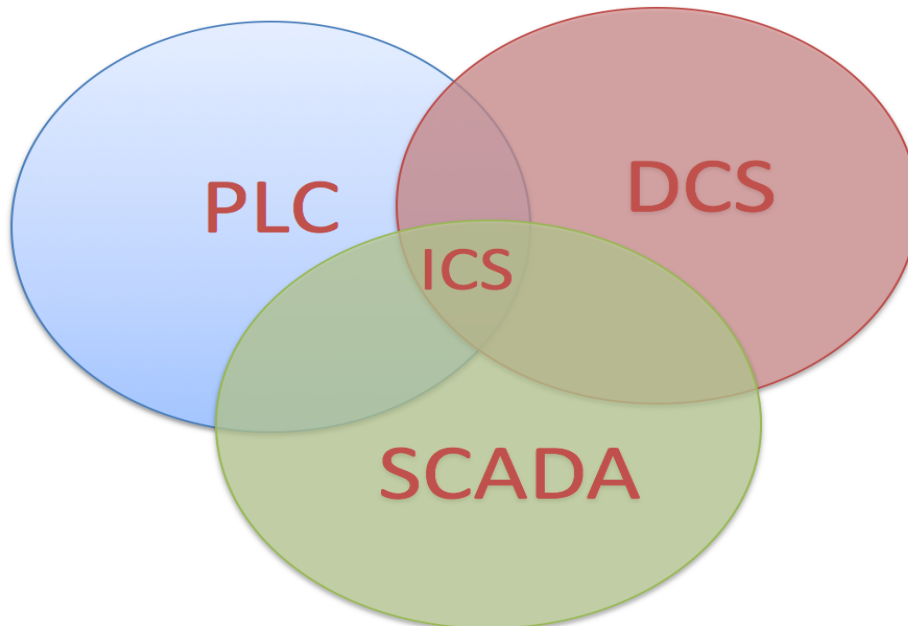


Figure 22-1: Industrial Control Systems (ICS)

Industrial Control Systems, like PLC (Programmable Logic Controller), DCS (Distributed Control System) and SCADA (Supervisory Control And Data Acquisition) share many of the same features.

The SCADA system typically contains different modules, such as:

1. OPC Server
2. A Database that stores all the necessary data
3. Control System
4. Datalogging System
5. Alarm System

They are typically implemented as separate applications because they should be able to run on different computers in a network (distributed). See Figure 22-2.

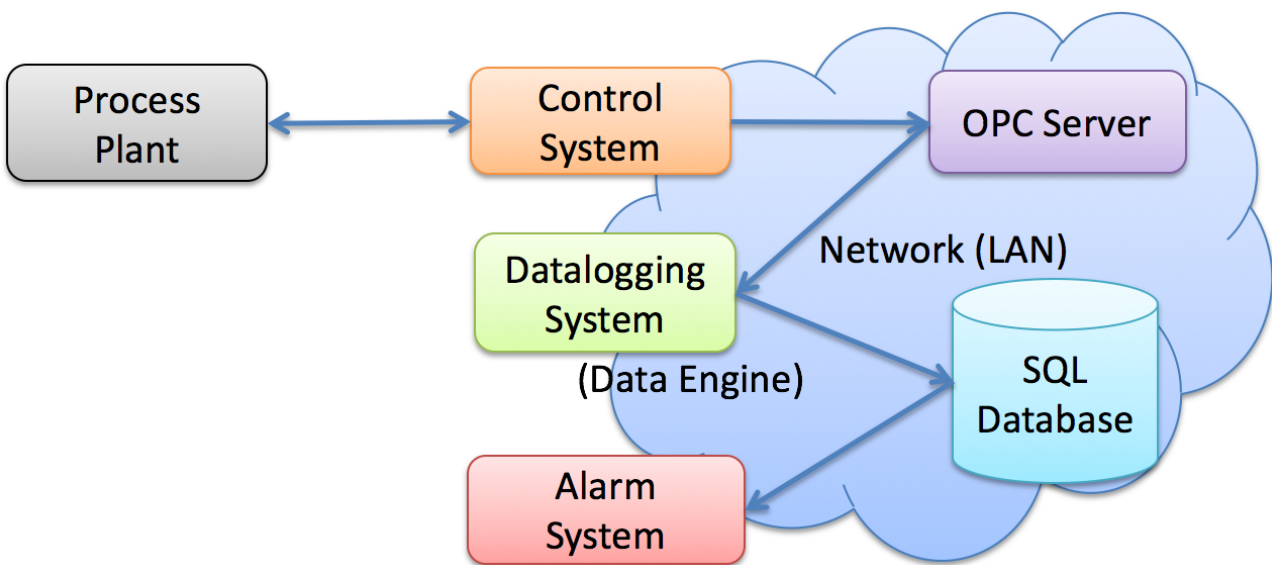


Figure 22-2: SCADA System Overview

For more details, see:

<https://www.halvorsen.blog/documents/technology/scada/>

23 HIL Simulation

Web: <https://www.halvorsen.blog/documents/technology/hil/>

23.1 What is HIL Simulation?

What is Hardware-in-the-Loop (HIL) Simulation or What is Hardware-in-the-Loop (HIL) Test?

The Hardware-in-the-Loop process has existed for no more than 15 to 20 years. Its roots are found in the Aviation industry. The reason the use of a HIL process is becoming more prevalent in all industries is driven by two major factors: time to market and complexity.

Hardware-in-the-loop (**HIL**) simulation is a technique that is used in the development and test of complex process systems. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform. The complexity of the plant under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the “plant simulation.”

Hardware-In-the-Loop is a form of real-time simulation. Hardware-In-the-Loop differs from real-time simulation by the addition of a real component in the loop. This component may be an “Electronic Control Unit” (ECU).

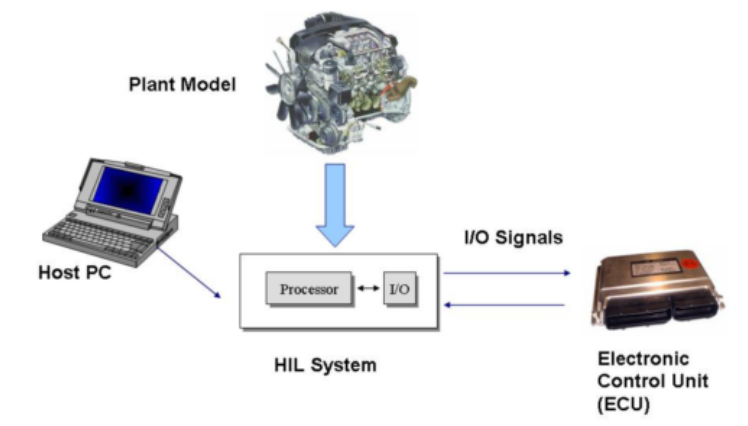


Figure 23-1: HIL Simulation

Figure 23-1 shows that the plant is simulated and the ECU is real. The purpose of a Hardware-In-the-Loop system is to provide all of the electrical stimuli needed to fully exercise the ECU. In this way, you “fool” the ECU into thinking that it is indeed connected to a real plant.

The HIL simulation includes a mathematical model of the process and a hardware device/ECU you want to test, e.g. an industrial PID controller we will use in our example. The hardware device is normally an embedded system.

23.2 Why use HIL simulation?

This question is an important part of understanding real-time technology. To restate the question using a control systems term: Why not connect the embedded system under test to the “real plant”, that is the dynamic system being controlled, to perform development and testing? In many cases, the most effective way to develop an embedded system is to connect the embedded system to the real plant, if such a plant exists. Increasingly however, HIL simulation is more efficient and or required.

The main purpose with the HIL Simulation is to test the hardware device on a simulator before we implement it on the real process.

The metric of development and test efficiency is typically a formula that includes the following factors:

- Cost
- Duration
- Safety

You may want to test the different part of the system individually to make sure it works as planned and HIL simulation is important in design and testing of the different systems.

It may be very useful, e.g., to test a controller function with a simulated process before the controller is applied to the real (physical) process. If the mathematical model used in the simulator is an accurate representation of the real process, you may even tune the controller parameters (e.g. the PID parameters) using the simulator.

It is also very useful for training purposes, i.e., the process operator may learn how the system works and operate by using the hardware-in-the-loop simulation.

Another benefit of Hardware-In-the-Loop is that testing can be done without damaging equipment or endangering lives. For instance, potentially damaging conditions in an engine, such as over-temperature, can be simulated to test if the ECU can detect and report it. Another instance would be an anti-lock braking (ABS) simulation at performance extremes. If simulated, the performance of the ABS system can be evaluated without risk to the vehicle or operator.

In Figure 23-2 we see a typical HIL test system:

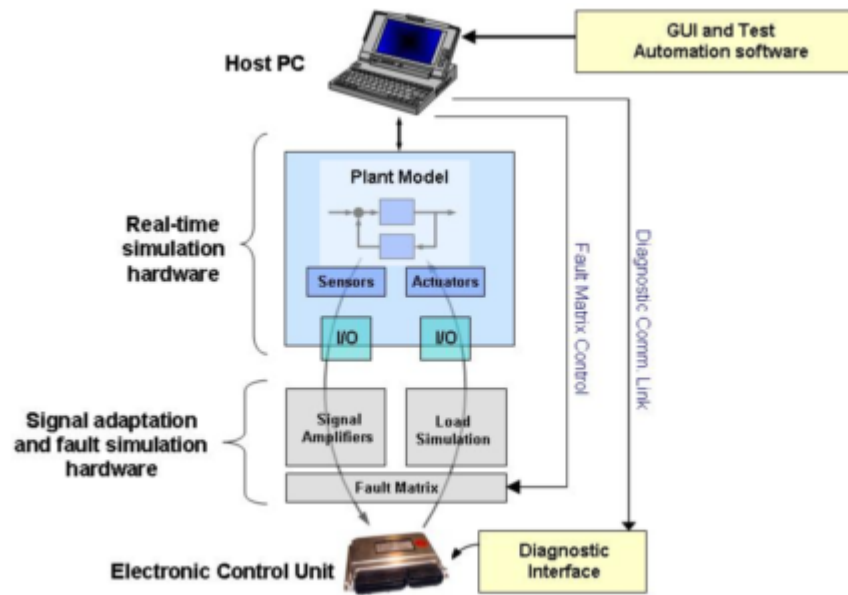


Figure 23-2: HIL Test System

HIL should be an integrated part of the design and testing cycle. The Figure below represents the design cycle of a typical system, e.g. a control system.

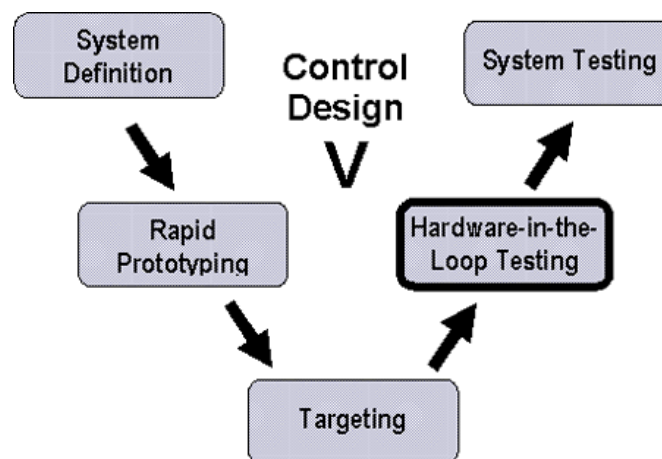


Figure 23-3: Design Cycle

As the complexity of the hardware being controlled increases, so too does the complexity of the embedded system that is designed to control the hardware. Hardware-in-the-Loop (HIL) simulation is a technique that is used increasingly in the development and test of complex real-time embedded systems.

The purpose of HIL simulation is to provide an effective platform for developing and testing real-time embedded systems, often in close parallel with the development of the hardware. Software development no longer needs to wait for a physical plant in order to write and test code.

HIL simulation provides an effective platform by adding the complexity of the plant under control to the development and test platform. The complexity of the plant under control is included in test

and development by adding a mathematical representation (model) of all related dynamic systems. These mathematical representations are referred to as the “plant simulation.”

23.3 Challenges

When testing, we have lots of challenges:

- Cost to test
- Cost of failure
- Availability
- System variation
- Repeatability

In these situations, is HIL simulation a powerful technique. With HIL Testing we will reduce cost and risk.

With HIL Testing cost and risk will be reduced:

- Increased reliability and quality
- More efficient development
- Lower cost to innovate

23.4 Applications

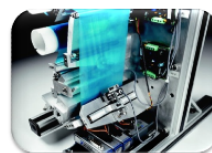
23.4.1 Embedded Control Systems

HIL simulation is widely used in developing Embedded Control Systems, such as:

- Medical Devices
- Industrial machines
- Power Generation Systems
- White Goods
- Aerospace
- Automotive
- Process Control



Medical Devices



Industrial Machines



Power Generation Systems



White Goods



Aerospace



Automotive

23.5 Procedure

The main steps in HIL Simulation are as follows:

1. Develop a mathematical model. Create a mathematical model of the real environment where the hardware device is meant to be used.
2. HIL Simulation (Software + Hardware). Test your device on a simulated process (mathematical model).
3. Implement your hardware on the Real Process (Hardware only). If everything is OK, you may want to implement your hardware device in the real environment where it meant to be used.

These tasks follow the main idea with a HIL simulation. First step is to simulate your system in software. Next is to test your hardware on the simulated process. Finally, you implement your hardware on the real system.

23.6 Practical Example

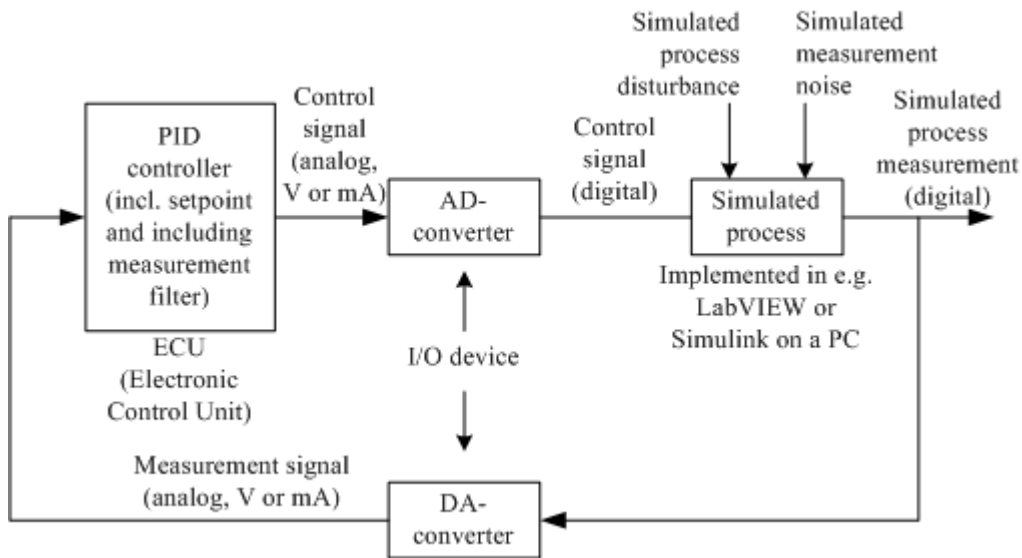
23.6.1 Introduction

It may be very useful to test a controller function with a simulated process before the controller is applied to the real (physical) process. If the mathematical model used in the simulator is an accurate representation of the real process, you may even tune the controller parameters (e.g. the PID parameters) using the simulator.

If the controller to be tested is implemented in the controller hardware, often denoted the electronic control unit (ECU), and the simulator has to run in real time, i.e. the simulation time develops as real time. This real-time simulation is obtained by setting the simulation algorithm cycle time equal to the simulation time step.

Typically, the simulator communicates with the ECU via ordinary I/O (current, voltage, digital). Such a system - where the real controller is controlling a simulated process - is denoted Hardware-in-the-loop (HIL) simulation. HIL-simulation is used in many industries, e.g. automotive industry for testing clutch automation systems and in marine and aircraft industry to test autopilots of vessels.

The Figure below illustrates the principle of testing a control system by replacing the physical system (or process) to be controlled by a simulated system. The controller is assumed to be a PID controller, but the figure applies to any controller function.



23.6.2 Simulated Process

In this example, a mathematical model of the following small-scale process is used (“Air Heater”):

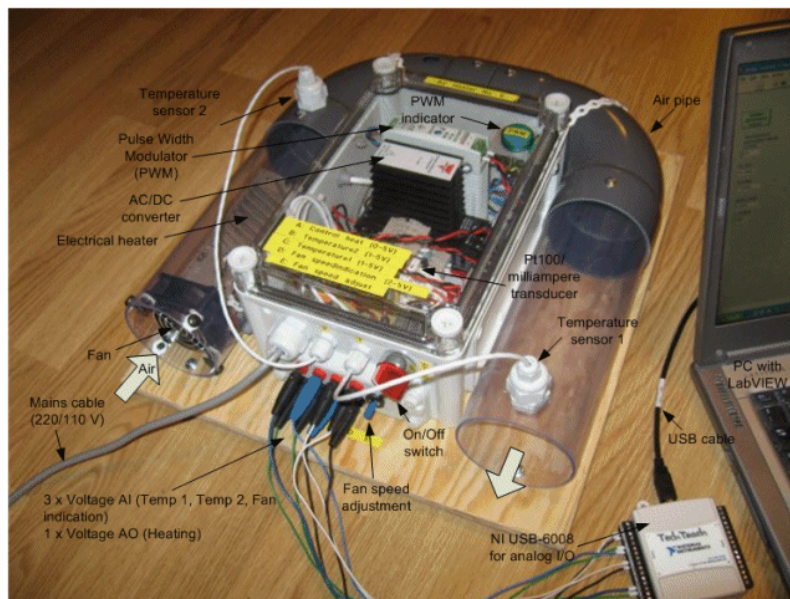


Figure 23-4: Air Heater System

The mathematical model is:

$$\dot{T}_{out} = \frac{1}{\theta_t} \{-T_{out} + [K_h u(t - \theta_d) + T_{env}]\}$$

Where:

- u [V] is the control signal to the heater.
- θ_t [s] is time-constant.
- K_h [deg C / V] is the heater gain.

- θ_d [s] is the time-delay representing air transportation and sluggishness in the heater.
- T_{env} is the environmental (room) temperature. It is the temperature in the outlet air of the air tube when the control signal to the heater has been set to zero for relatively long time (some minutes).

23.6.3 Hardware

The main purpose with the HIL Simulation is to test the hardware device on a simulator before we implement it on the real process.

In this we use an ordinary industrial PID controller, such as Fuji PGX5.

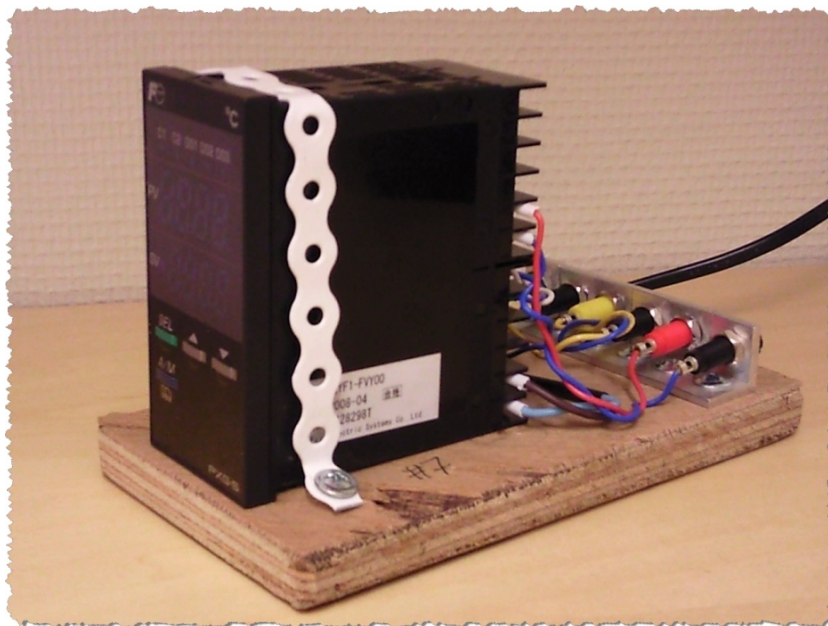


Figure 23-5: Fuji PGX5 PID Controller

We will test the **Fuji PGX5 PID** controller on a model, and if everything is OK we will implement the controller on the real system.

We will use **LabVIEW** in order to implement the HIL Simulation. LabVIEW is a graphical programming language from Nation Instruments, and it is well suited for such implementation.

23.6.4 The Procedure

The procedure is as follows:

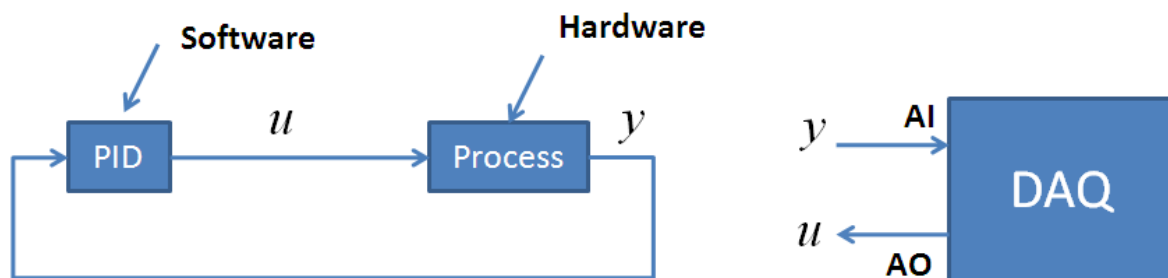
4. **PID Control and Simulation in LabVIEW (Software only)**. Simulate the model and implement the built-in PID controller in LabVIEW. No hardware involved.
5. **Configure the Fuji PGX5 PID controller (Hardware only)**. Configure and be familiar with the industrial Fuji PGX5 PID controller.

6. **HIL Simulation in LabVIEW (Software + Hardware).** Test your industrial Fuji PGX5 PID controller on your simulated process.
7. **PID Tuning (Software + Hardware).** Find proper PID parameters, etc. for the controller based on the model.
8. **Implement your hardware, i.e., the Fuji PGX5 PID controller on the Real Process (Hardware only).** Now that you have tested your Fuji PGX5 PID controller on the simulated process, it's time to implement it on the real process. Fine-tune PID parameters if necessary.

These tasks follow the main idea with a HIL simulation. First step is to simulate your system in software. Next is to test your hardware on the simulated process. Finally you implement your hardware on the real system.

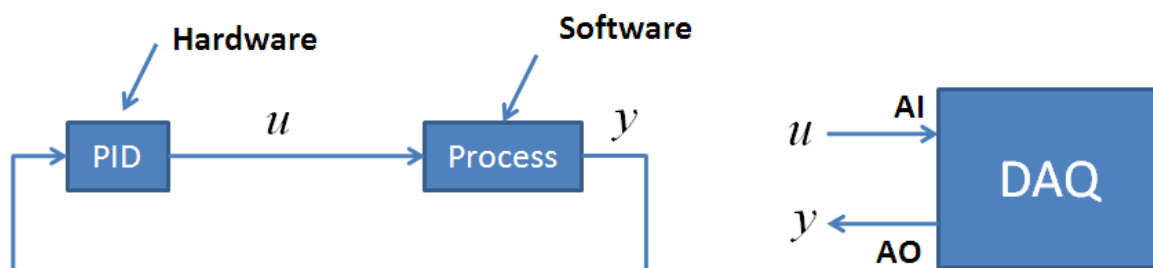
Below we see the difference between a traditional process system using a software program for implementing the control system and a HIL simulation.

Traditional process system using a software program for implementing the control system:



In this case you need to scale the voltage signal you get from the process and the DAQ to a temperature value ($1 - 5V \rightarrow 20 - 50^{\circ}C$).

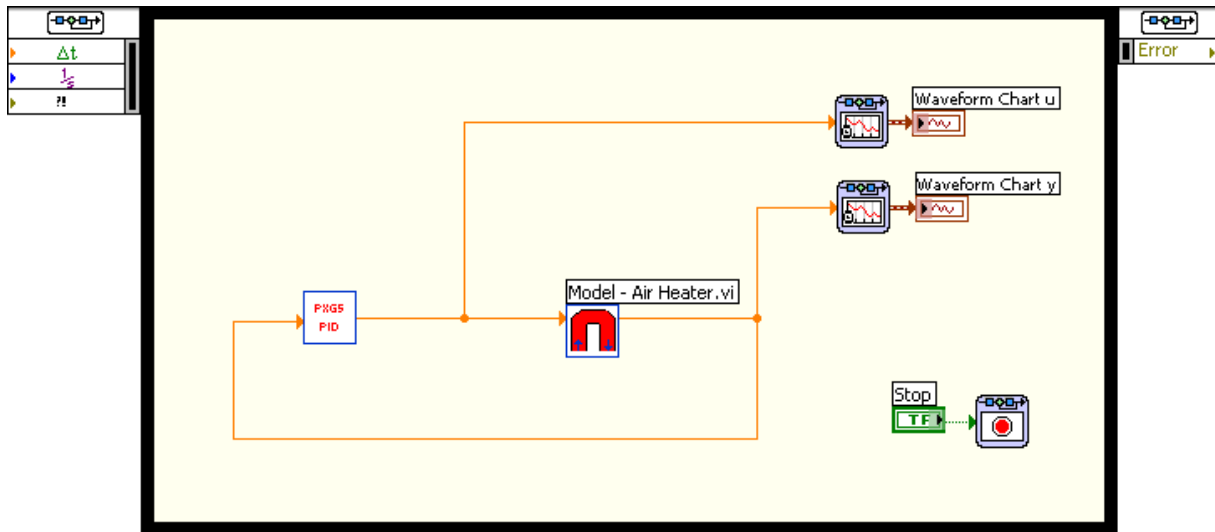
HIL Simulation:



In this case you need to scale the temperature value you get from the simulated process before you send the value to the Fuji PGX5 PID controller ($20 - 50^{\circ}C \rightarrow 1 - 5V$).

23.6.5 HIL Simulation in LabVIEW

Below we see an excerpt of the program created in LabVIEW:

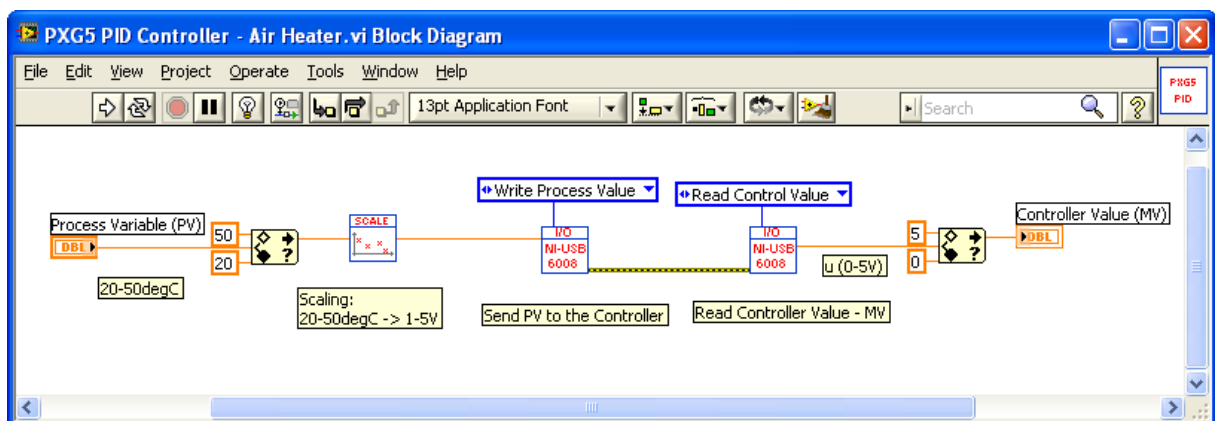


In the example we have used a “Simulation Loop” in LabVIEW, but an ordinary While Loop may also be used. The model is implemented in a Simulation Subsystem.

PXG5 PID.vi:

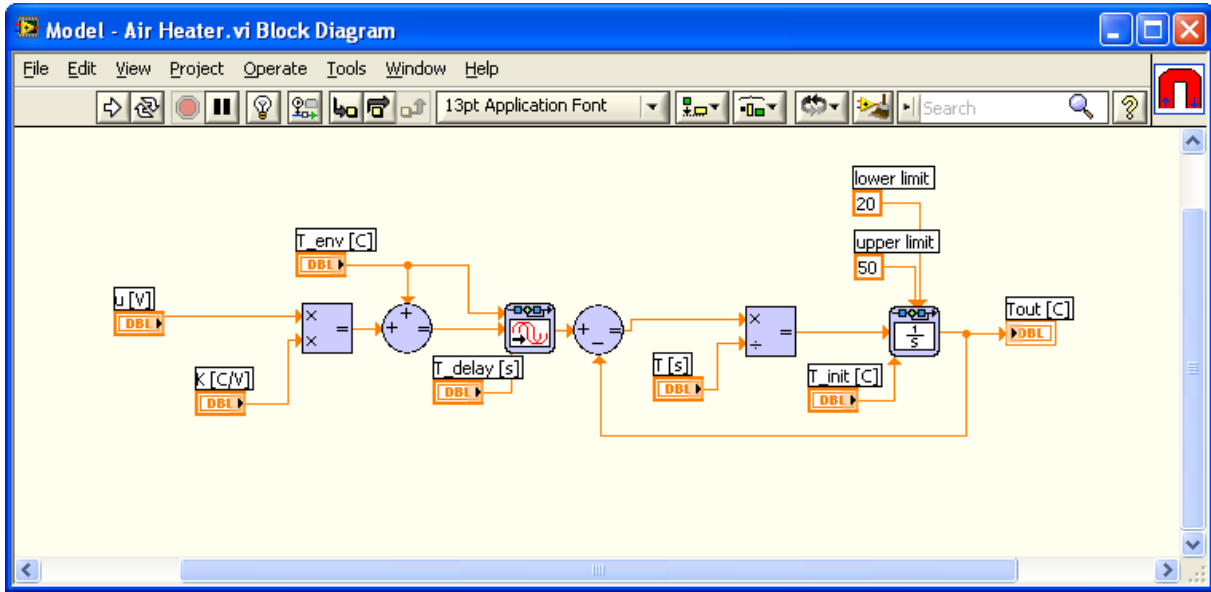
Inside the SubVI “**PXG5 PID.vi**” is the I/O from and to the PXG5 PID controller implemented using an ordinary DAQ device (NI USB-6008 USB DAQ device), i.e., the simulated process value needs to be sent to the controller and the manipulated value from the controller need to be sent to the simulated process. Scaling is also implemented in this SubVI.

Below we see the “PXG5 PID.vi”:



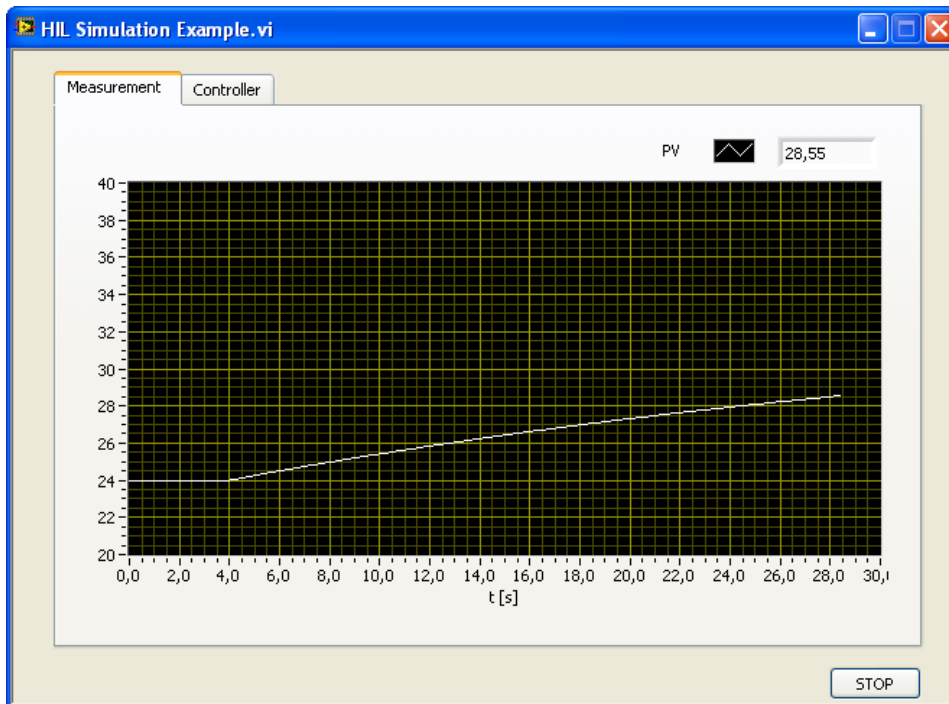
Mathematical Model:

In the Model – Air Heater.vi simulation subsystem is the mathematical model implemented as shown below:



Results:

The simulation results become:



The Set Point (SP) is set on the PXG5 PID controller (in this case 30°C at time $t = 2s$). The simulation is based on PID parameters set on the PXG5 PID controller using the built-in Auto-tuning functionality that the PXG5 PID controller has.

Part 4 : Internet of Things

In this part, we will give an overview of Internet of Things, Home Automation and devices such as Arduino and Raspberry Pi.



In Figure 24-2 we see a chart of how many devices that is already connected to Internet and how many devices possibly connected in the future.

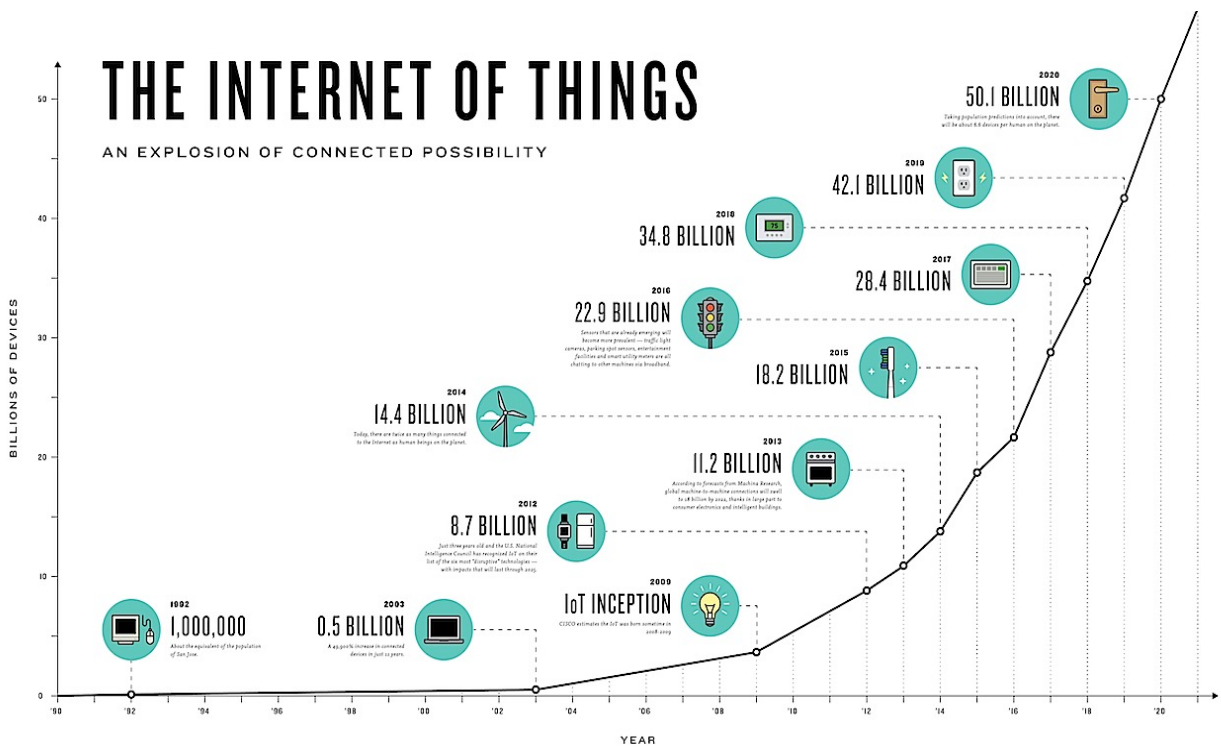


Figure 24-2: IoT – Number of Devices connected to Internet

The devices we will focus on in this document are Arduino and Raspberry Pi (Figure 24-3). These devices are small computers or microcontrollers. These devices will be explained in more detail later.

Internet of Things (IoT)

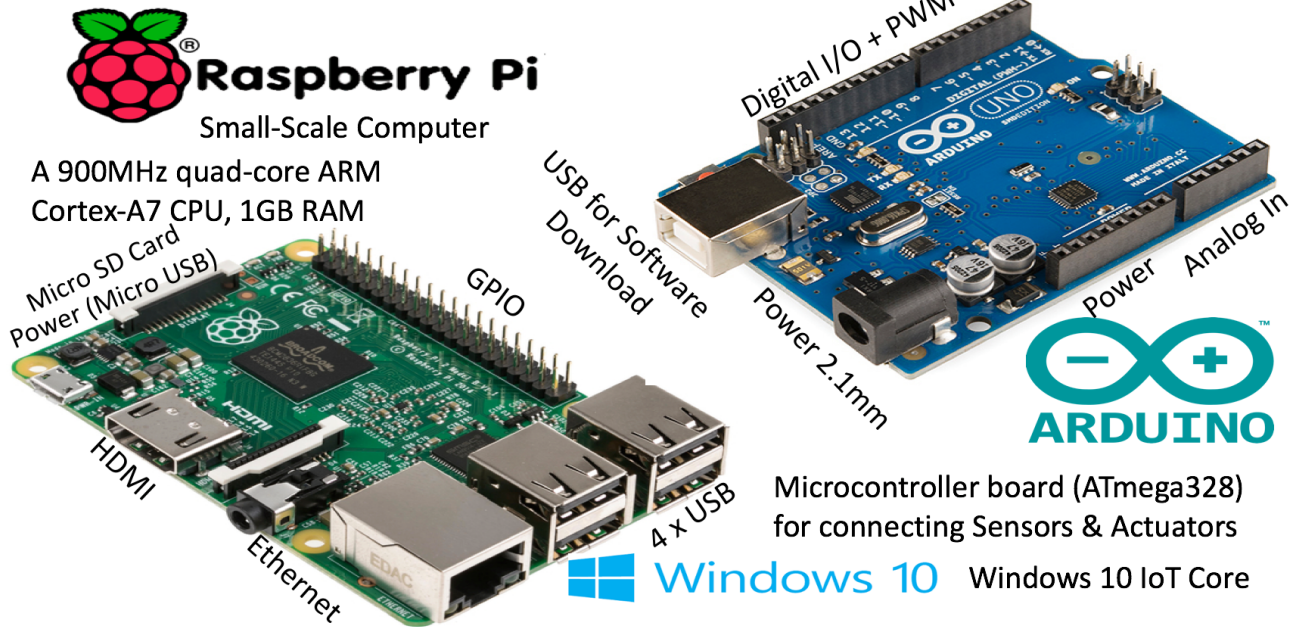


Figure 24-3: Internet of Things Devices

24.1 Data Logging

An important part of IoT is to log data from all these devices.

24.1.1 Web-based Logging Services

Examples:

- Temboo by LogMeIn
- Xively
- ThingSpeak
- Etc.

See Figure 24-4.

Web-based Logging Service

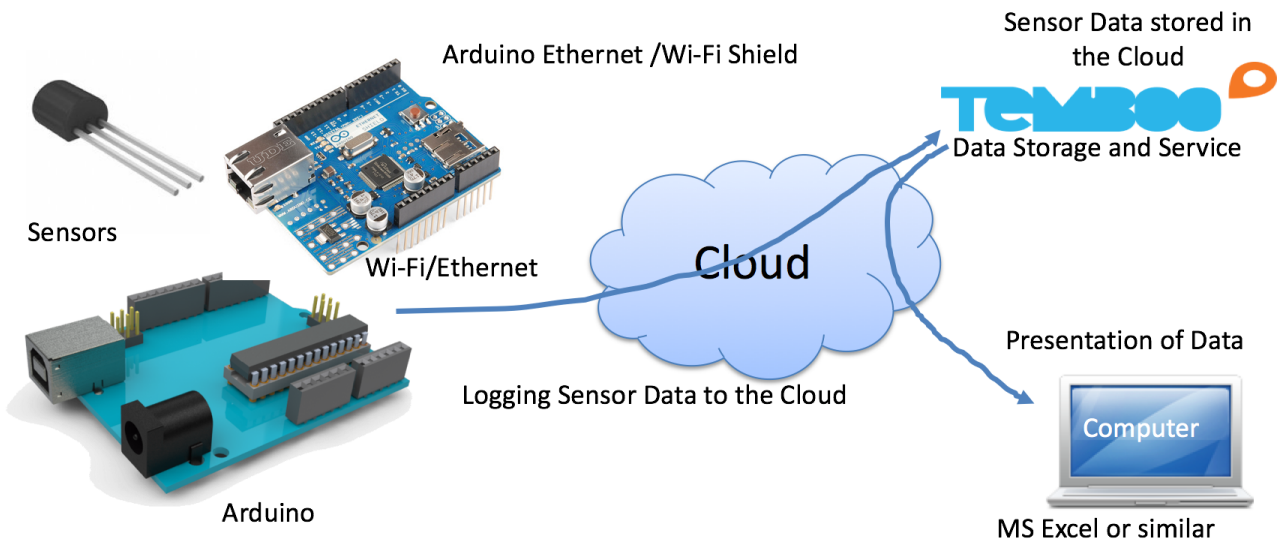


Figure 24-4: Web-based Logging Service Example

A lot of existing solutions do exist, here I will mention Temboo, Xively and ThingSpeak.

Temboo

www.temboo.com

Connect your Arduino & Arduino-compatible devices to a vast array of web-based resources and services with the power of Temboo.

Xively by LogMeIn

<https://xively.com>

<https://en.wikipedia.org/wiki/Xively>

Xively by LogMeIn offers an Internet of Things (IoT) platform as a service, business services, and partners that enable businesses to quickly connect products and operations to the Internet.

ThingSpeak:

ThingSpeak is a IoT Cloud Service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.

It works with Arduino, Raspberry Pi and MATLAB, etc.

<https://thingspeak.com>

25 Home Automation

Home automation (also known as Smart House, Smart Home, etc.) solutions has greatly increased in popularity over the past several years. Home Automation may include centralized control of lighting, heating, ventilation and air conditioning, appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security.

https://www.halvorsen.blog/documents/projects/projects/smart_buildings.php

Figure 23-1 shows some examples.

Home Automation Examples



Security and Surveillance



Temperature Control



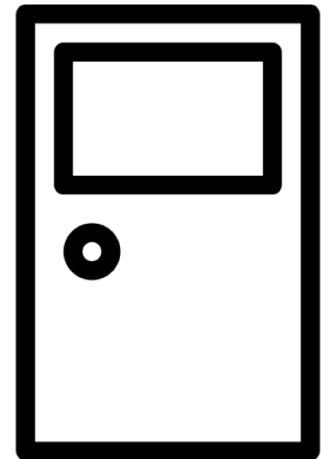
Light Control



Temperature Measurements

Indoor and Outdoor (Measurements and Predictions)

Day/Night Control



Access Control

Figure 25-1: Home Automation Examples

In Figure 25-2 we see a typical house with Home Automation.

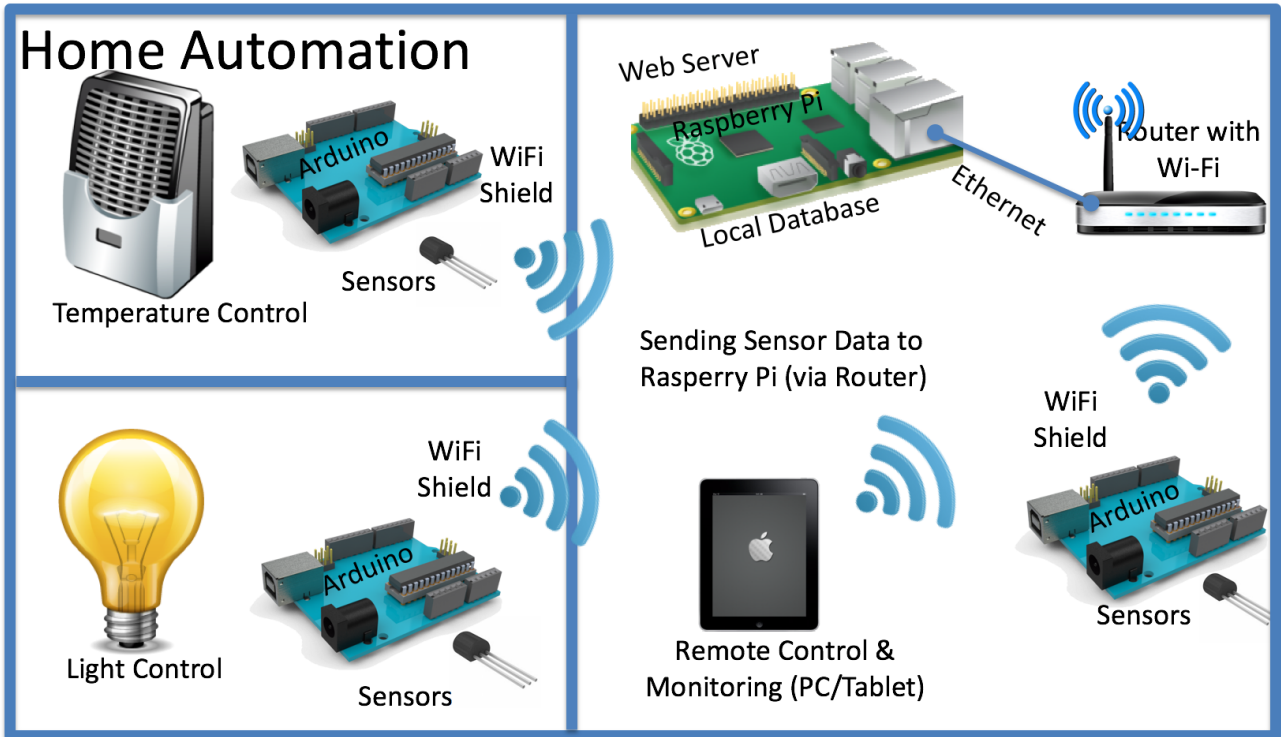


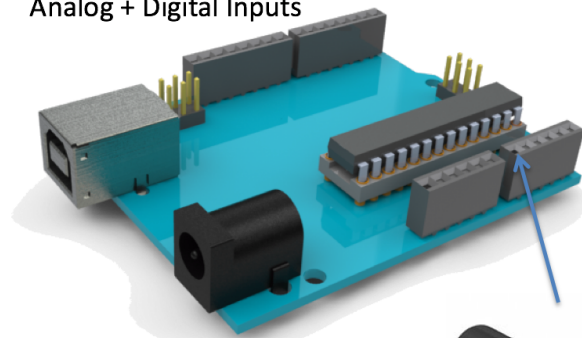
Figure 25-2: A Typical House with Home Automation

In this example, we are using Arduino and Raspberry Pi as monitoring and control devices, see Figure 25-3.

Home Automation Devices

Arduino

Analog + Digital Inputs

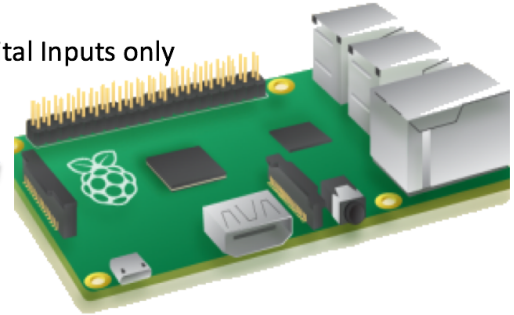


I/O Module for Connecting Sensors & Actuators

Sensors

Raspberry Pi

Digital Inputs only



Small-scale Computer with Possible Database & Web Server

Figure 25-3: Home Automation with Arduino and Raspberry Pi

25.1 Home Automation Platform

In this subchapter, we will give an overview of a Home Automation platform.

Figure 25-4 shows an overview of the platform.

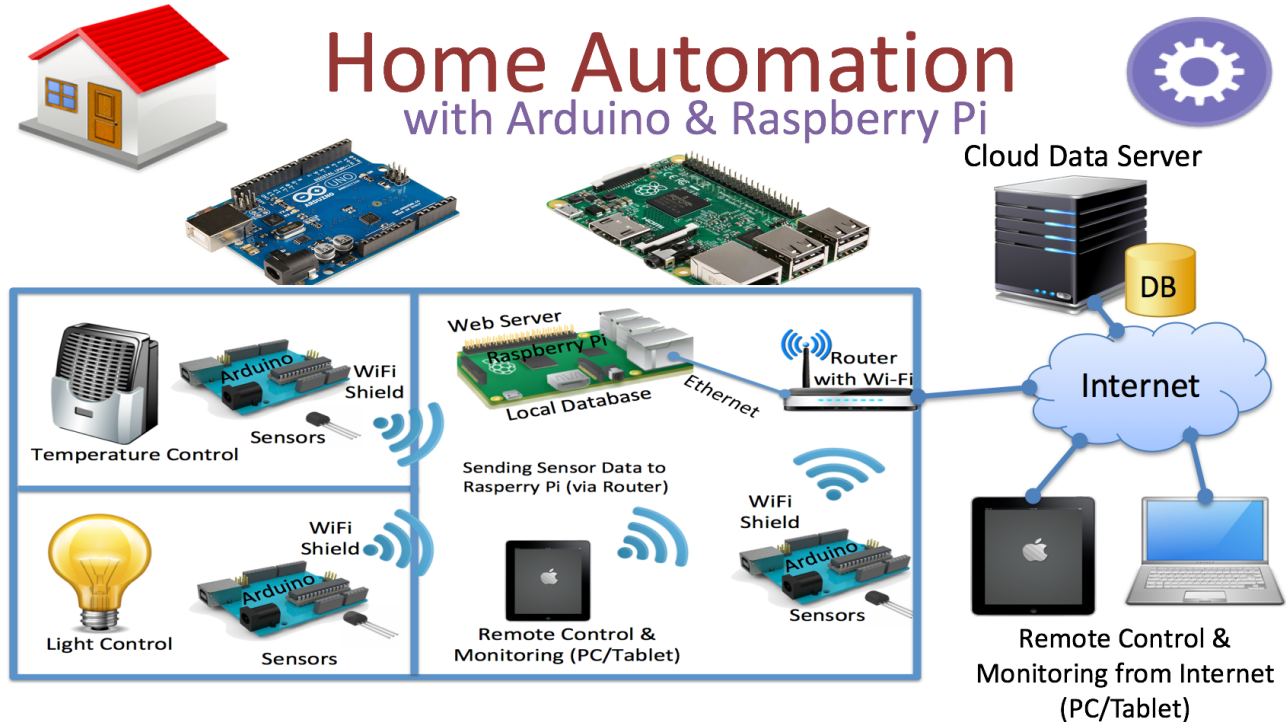


Figure 25-4: Home Automation Platform

The platform consists of 2 main parts:

- Data Management
- Data Monitoring

These parts will be explained below.

Management:

Here we can configure Devices and Tags (Figure 25-5, Figure 25-6)

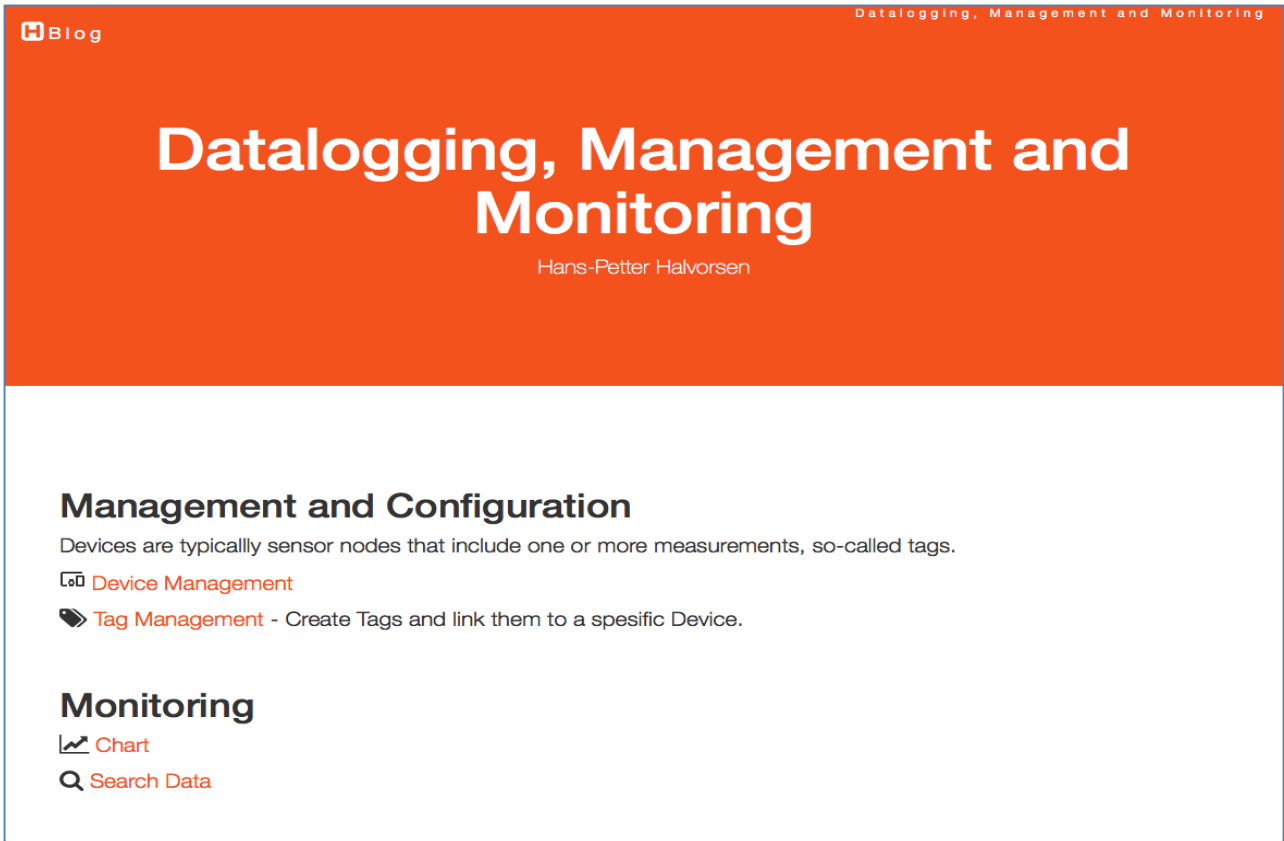


Figure 25-5: DMM Platform - Management

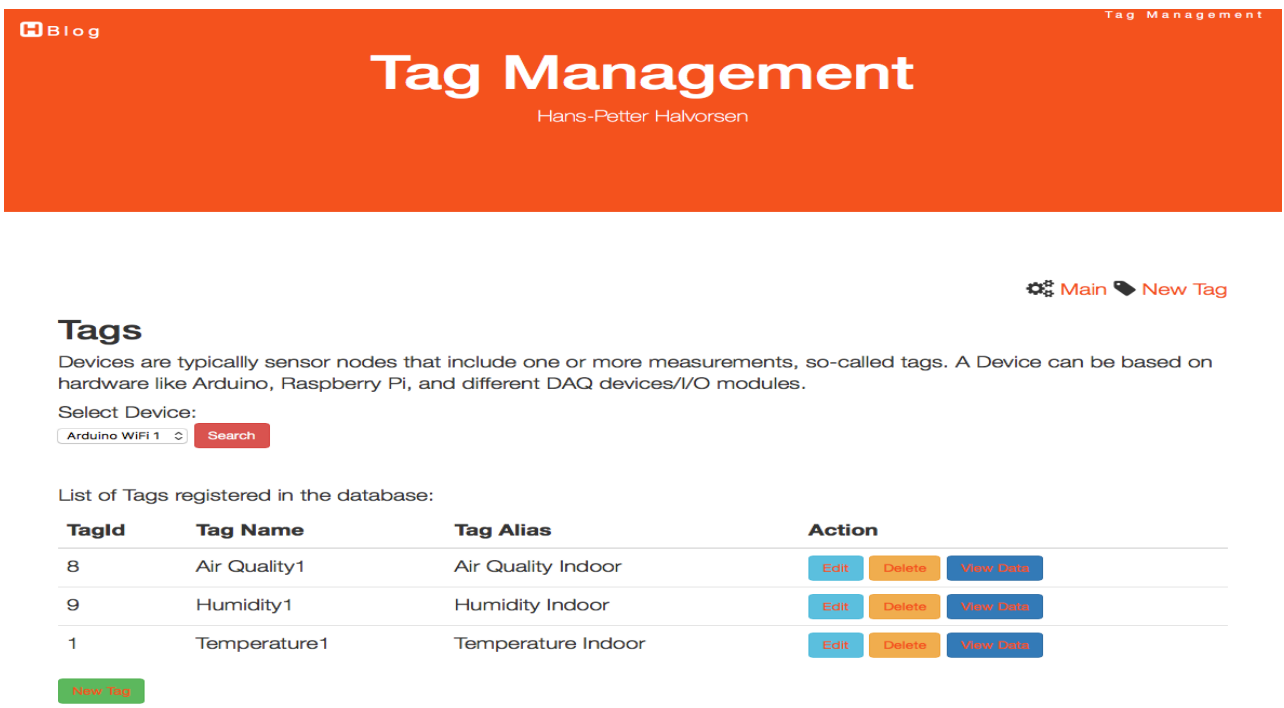


Figure 25-6: DMM Management – Tag Configuration

Monitoring:

Here (Figure 25-7) we can see the logged data with charting possibilities, etc.

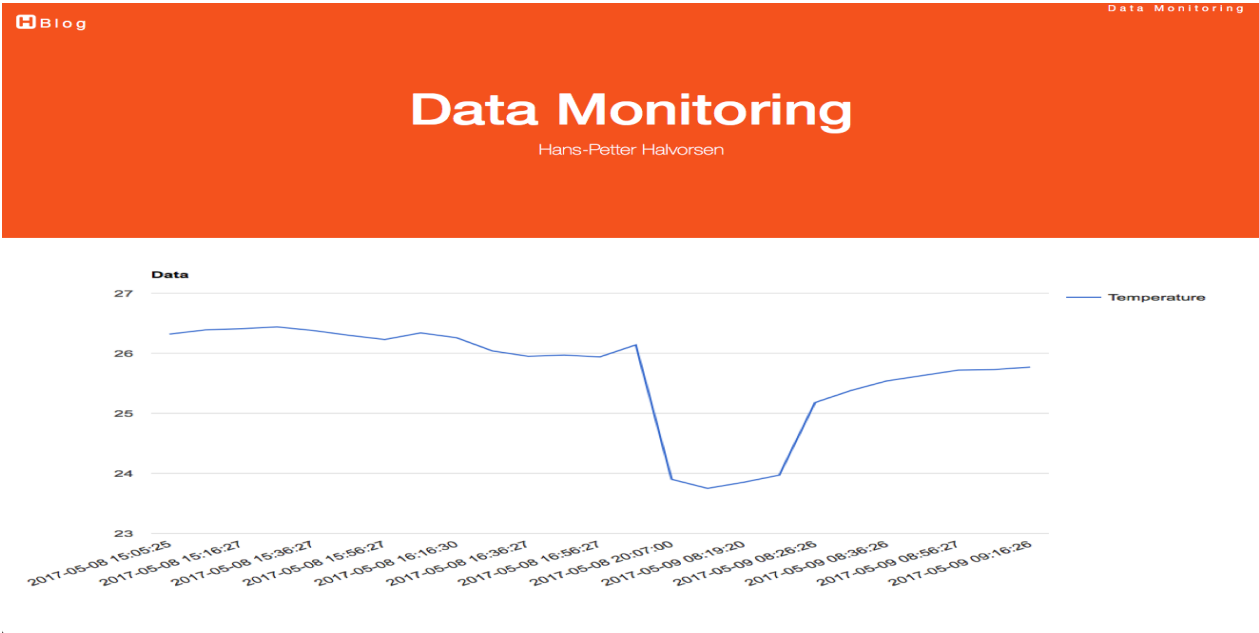


Figure 25-7: DMM Platform - Monitoring

With the DMM platform you can manage and monitor IoT data for one or many houses.

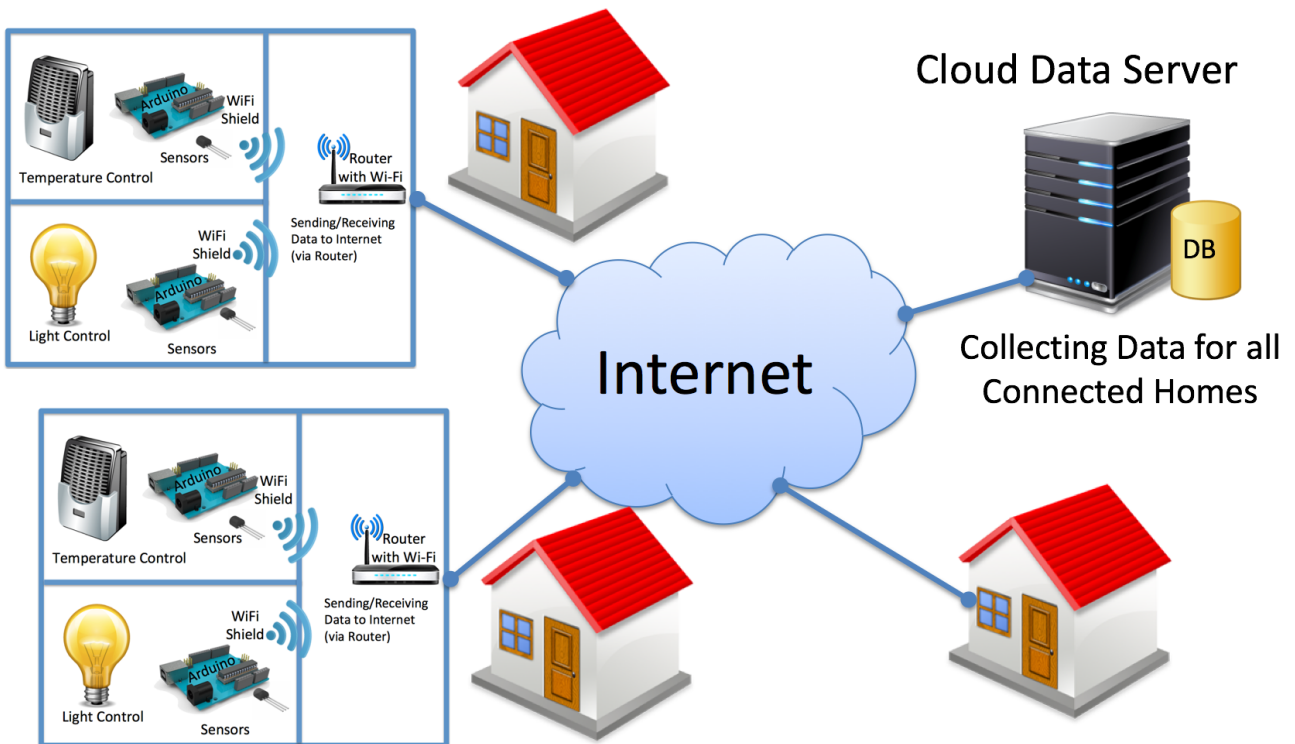


Figure 25-8: DMM Overview

For more information, refer to the following web site:

<https://www.halvorsen.blog/documents/projects/projects/dmm.php>

26 Arduino

Web: <https://www.halvorsen.blog/documents/technology/iot/arduino.php>

Arduino (Figure 26-1) is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, and turn it into an output - activating a motor, turning on an LED, etc. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language, and the Arduino Integrated Software Environment (IDE).

www.arduino.cc

<https://en.wikipedia.org/wiki/Arduino>

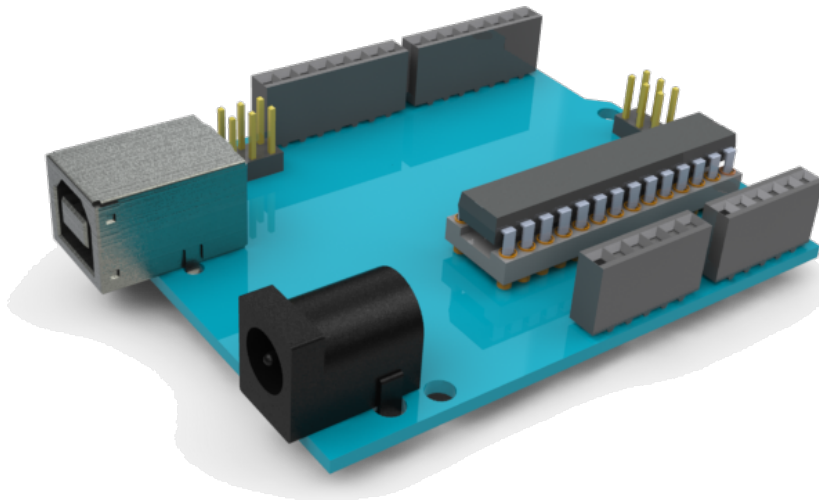


Figure 26-1: Arduino

Lots of different Arduino boards do exist, we will focus on the most popular board, namely Arduino UNO.

For information about other boards, please see:

<https://www.arduino.cc/en/Main/Products>

26.1 Arduino UNO

The Uno is a microcontroller board. It has 14 digital input/output pins (of which 6 can be used as PWM outputs) and 6 analog inputs. The operating voltage is 5V.

Arduino

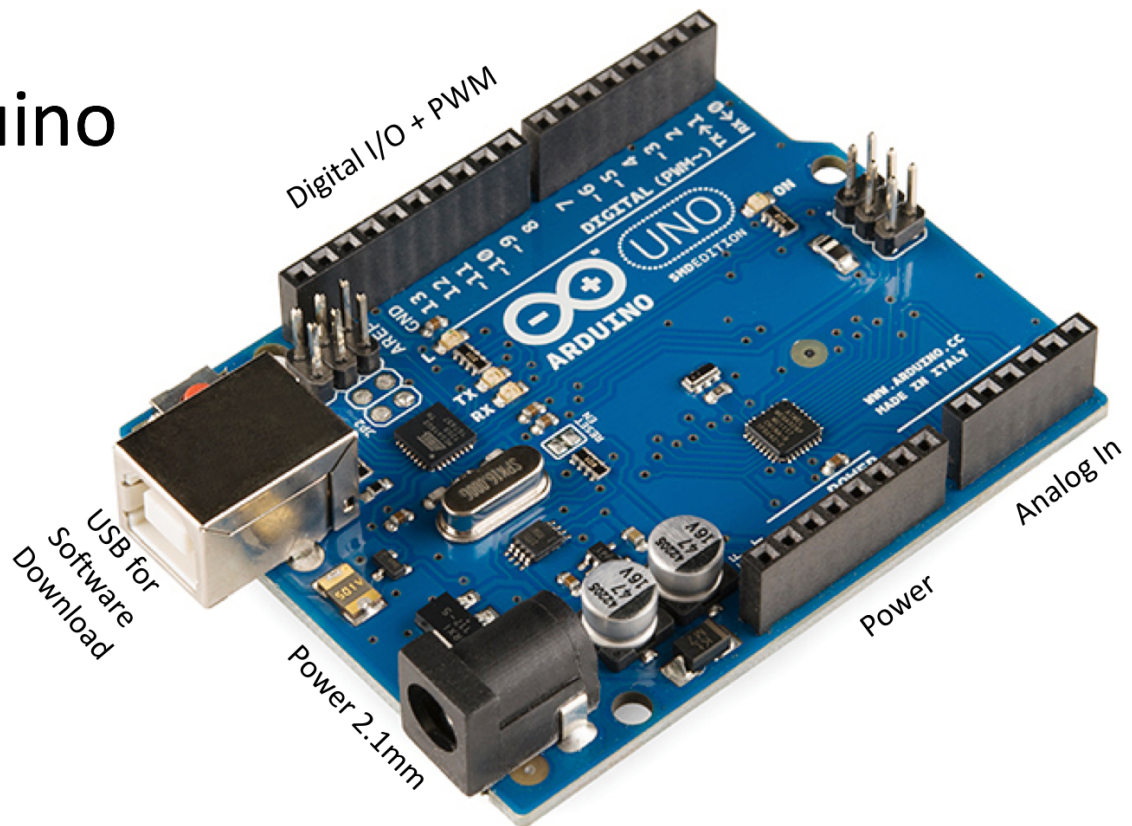


Figure 26-2: Arduino UNO

For more information about Arduino UNO, see the following:

<https://www.arduino.cc/en/Main/ArduinoBoardUno>

26.2 Sensors and Actuators

Figure 26-3 shows some typical sensors and actuators we can use with Arduino.

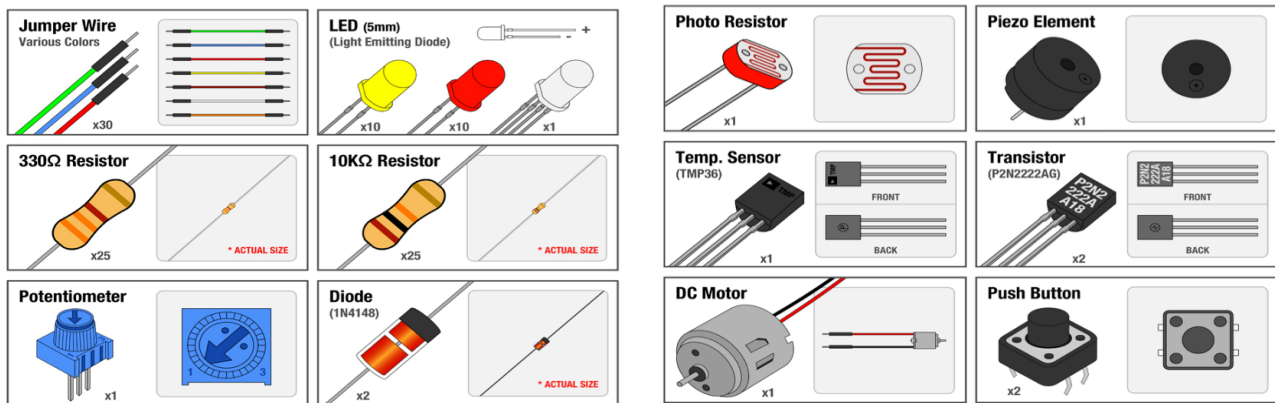


Figure 26-3: Typical Sensors and Actuators used with Arduino

Arduino Starter Kit:

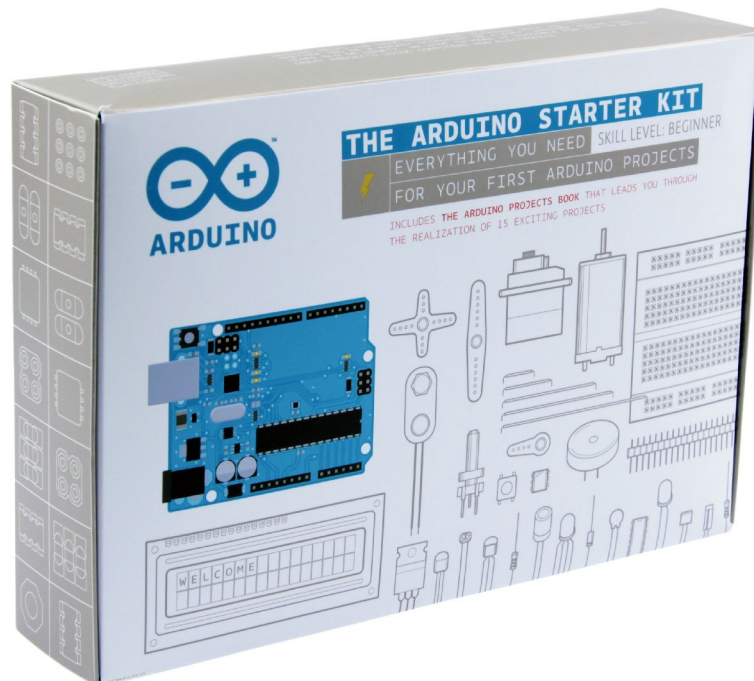


Figure 26-4: The Arduino Starter Kit

Read more about the Arduino Starter Kit here:

<http://arduino.cc/en/Main/ArduinoStarterKit>

Starter Kit Videos:

https://www.youtube.com/playlist?feature=edit_ok&list=PLT6rF_I5kknPf2qIVFlvH47qHvqvzkknd

26.3 Software

Software Installation: <http://arduino.cc/en/Main/Software>

In Figure 26-5 we see the programming environment (IDE) for Arduino.

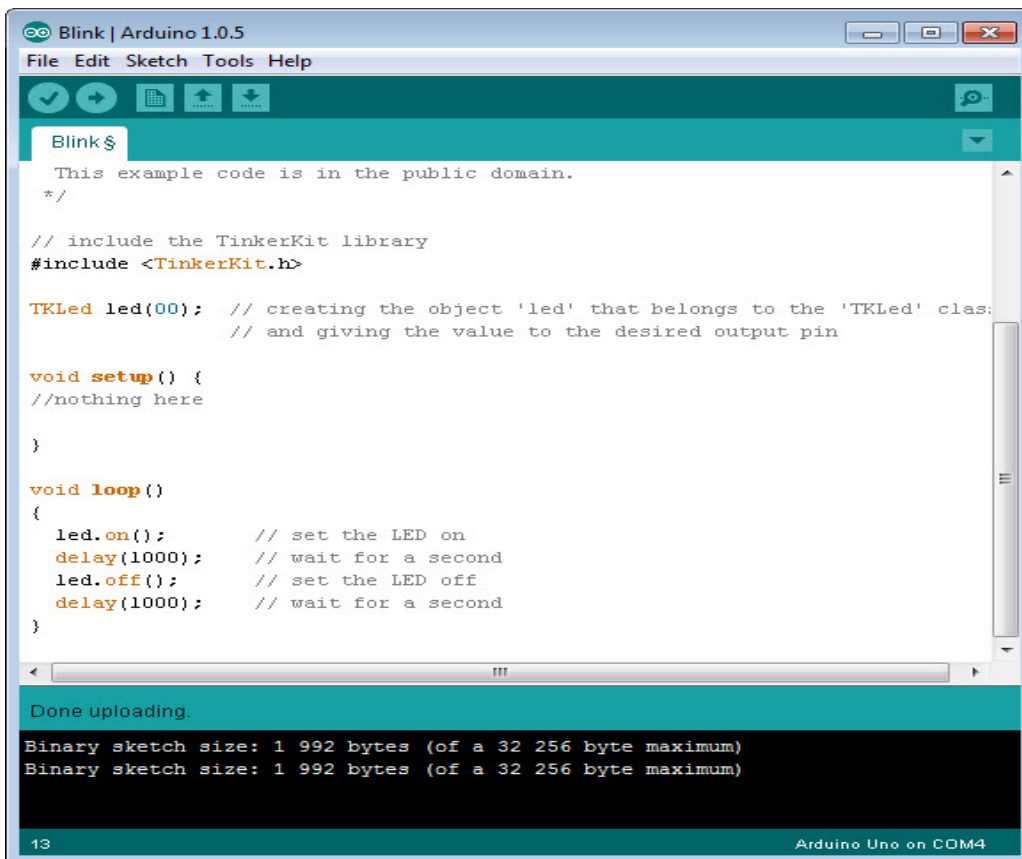


Figure 26-5: Arduino Programming Environment (Sketch)

The syntax is similar to C programming.

Example:

This example shows the simplest thing you can do with an Arduino to see physical output: it blinks an LED (Figure 26-6).



Figure 26-6: LED

Hardware Required

- Arduino Board, e.g., Arduino UNO

- LED (any color)
- 220Ω resistor

The value of the resistor in series with the LED may be of a different value than 220Ω; the LED will lit up also with values up to 1kΩ.

In Figure 26-7 we see the wiring for how to connect a LED to the Arduino:

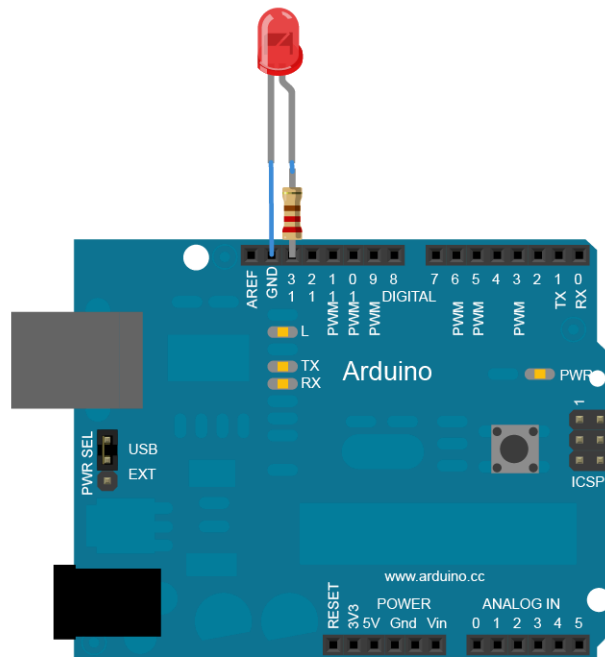


Figure 26-7: Connecting a LED to the Arduino

Sketch Program:

```
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

For more details, please see <https://www.arduino.cc/en/Tutorial/Blink>.

26.4 Code Examples

Below we will see some examples of how to use Arduino to read temperature values from different temperature sensors. The sensors used in the examples are very inexpensive and quite easy to use.

26.4.1 TMP36 Temperature Sensor Example

In this example we will use a TMP36 temperature sensor, see Figure 26-8.



Figure 26-8: TMP36 Temperature Sensor

Technical Data (Figure 26-9):

Technical data	
Temperature measurement range	-40...+125 °C
Accuracy	±2 °C (0...70 °C)
Power supply	2.3...5.5 V
Package	TO-92
Temperature sensitivity, voltage	10 mV/°C

Figure 26-9: TMP36 Technical Data

From the data sheet we also have the following plot (Figure 26-10):

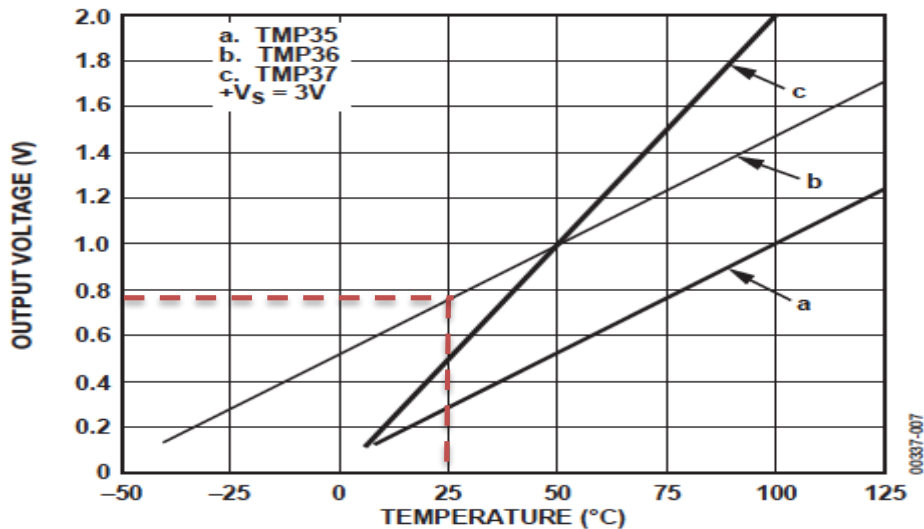


Figure 6. Output Voltage vs. Temperature

Figure 26-10: TMP36 Voltage vs. Temperature

We have a linear relationship:

$$y = ax + b$$

From the plot, we have:

$$(x_1, y_1) = (750\text{mV}, 25^\circ\text{C})$$

$$(x_2, y_2) = (1000\text{mV}, 50^\circ\text{C})$$

We then use the following formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

You have to find a (slope) and b (intercept):

$$y - 25^\circ\text{C} = ((50^\circ\text{C} - 25^\circ\text{C}) / (1000\text{mV} - 750\text{mV})) * (x - 750\text{mV})$$

This gives:

$$y[^\circ\text{C}] = (1/10) * x[\text{mv}] - 50$$

For more information about the sensor: <https://www.sparkfun.com/products/10988>

Wiring (Figure 26-11):

Arduino + Sensors

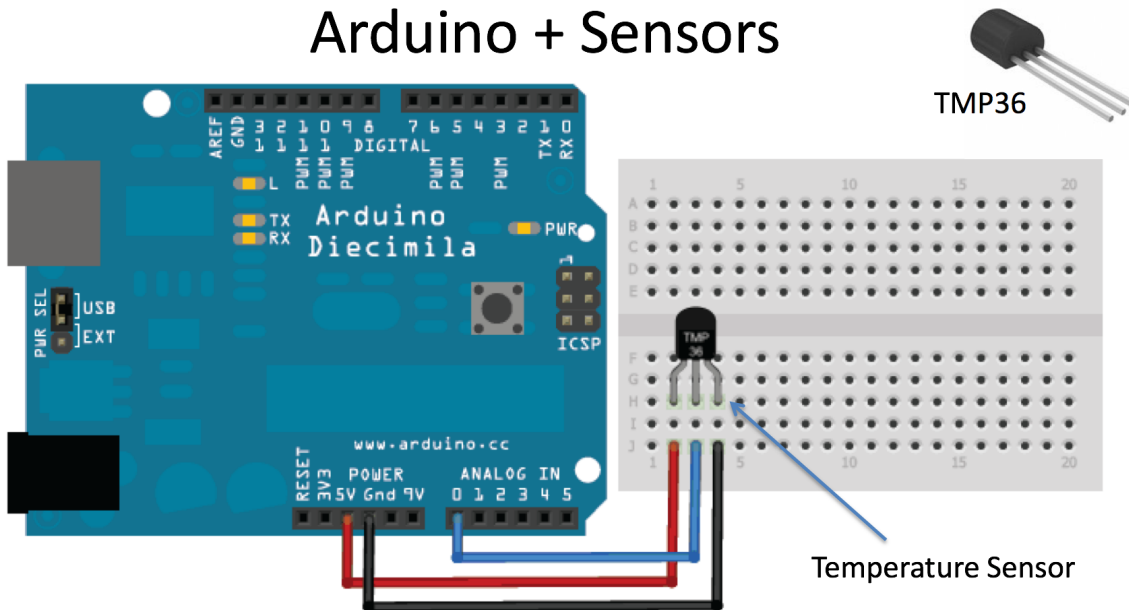


Figure 26-11: Hooking up Arduino with Sensors using a Breadboard

Sketch:

```

TMP36 Temperature Sensor Example
// We'll use analog input 0 to read Temperature Data
const int temperaturePin = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  float voltage, degreesC, degreesF;
  voltage = getVoltage(temperaturePin);

  // Now we'll convert the voltage to degrees Celsius.
  // This formula comes from the temperature sensor datasheet:
  degreesC = (voltage - 0.5) * 100.0;

  // Send data from the Arduino to the serial monitor window
  Serial.print("voltage: ");
  Serial.print(voltage);
  Serial.print(" deg C: ");
  Serial.println(degreesC);
  delay(1000); // repeat once per second (change as you wish!)
}
float getVoltage(int pin)
{
  return (analogRead(pin) * 0.004882814);

  // This equation converts the 0 to 1023 value that analogRead()
  // returns, into a 0.0 to 5.0 value that is the true voltage
  // being read at that pin.
}

```

Serial Monitor (Figure 26-12):

```

/dev/tty.usbmodem1421
voltage: 0.72 deg C: 21.78
voltage: 0.72 deg C: 21.78
voltage: 0.72 deg C: 21.78
voltage: 0.72 deg C: 21.78
voltage: 0.72 deg C: 21.78
voltage: 0.72 deg C: 21.78
voltage: 0.71 deg C: 21.29
voltage: 0.72 deg C: 21.78
voltage: 0.73 deg C: 22.75
voltage: 0.73 deg C: 23.24
voltage: 0.74 deg C: 23.73
voltage: 0.74 deg C: 24.22
voltage: 0.75 deg C: 25.20
voltage: 0.75 deg C: 25.20
voltage: 0.75 deg C: 24.71

```

Figure 26-12: Serial Monitor output from the TMP36 Example

26.4.2 NTC Thermistor Example

In this example, we will use a NTC thermistor temperature sensor, see Figure 26-8.



Figure 26-13: NTC Thermistor

NTC Thermistor Technical Data (Figure 26-14):

Technical data	
Resistance @ 25°C	10 kΩ
Temperature range	-40...+125 °C
Power max.	500 mW
Pitch	2.54 mm
Resistance tolerance	±5 %
W_{25/100} value	3977 K
B value tolerance	±0.75 %
Thermal time constant	15 s

Figure 26-14: NTC Thermistor Technical Data

Wiring (Figure 26-15):

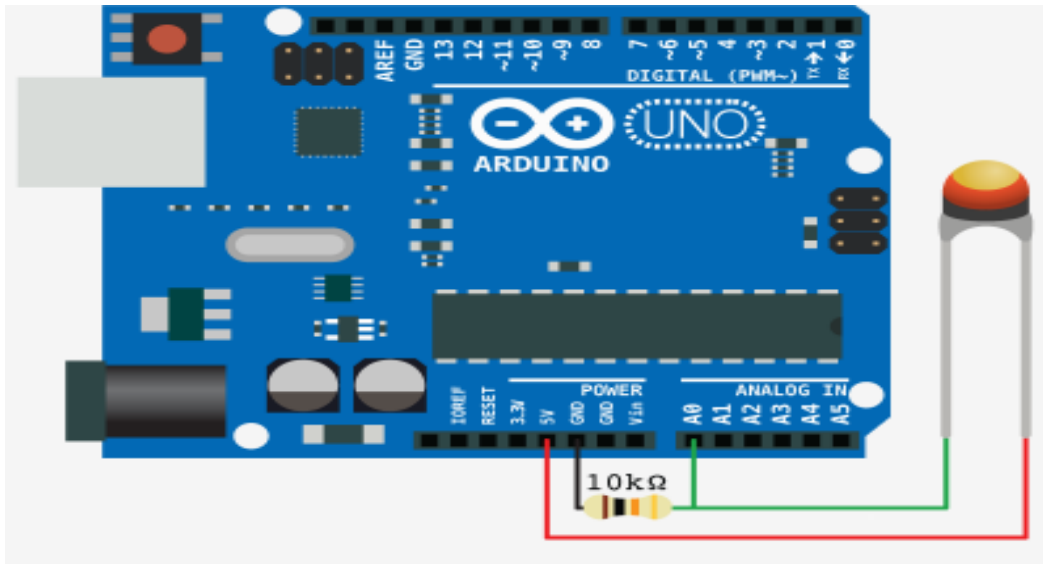


Figure 26-15: NTC Thermistor Wiring

The problem with resistance sensors is that the Arduino analog interfaces can't directly detect resistance changes.

This will require some extra electronic components. The easiest way to detect a change in resistance is to convert that change to a voltage change. You do that using a voltage divider (see wiring in Figure 26-15 and Figure 26-16).

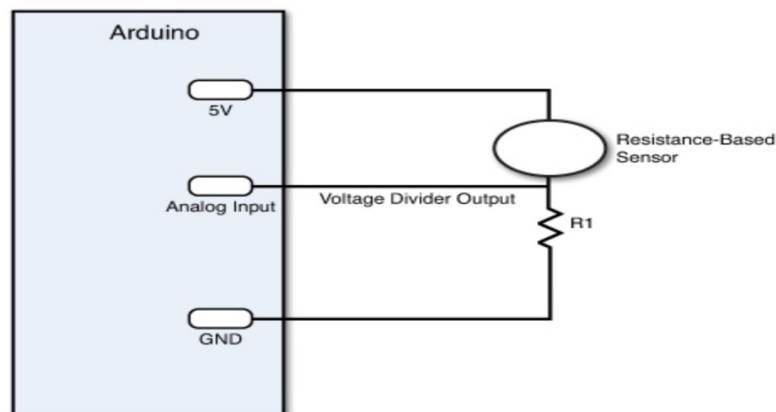


Figure 26-16: Voltage Divider

By keeping the power source output constant, as the resistance of the sensor changes, the voltage divider circuit changes, and the output voltage changes. The size of resistor you need for the R1 resistor depends on the resistance range generated by the sensor and how sensitive you want the output voltage to change.

Generally, a value between 1K and 10K ohms works just fine to create a meaningful output voltage that you can detect in your Arduino analog input interface.

We have used the Steinhart-Hart Equation in order to find the temperature:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$

Sketch:

```
// Read Temperature Values from NTC Thermistor
const int temperaturePin = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  int temperature = getTemp();
  Serial.print("Temperature Value: ");
  Serial.print(temperature);
  Serial.println("*C");
  delay(1000);
}
double getTemp()
{
  // Inputs ADC Value from Thermistor and outputs Temperature in Celsius
  int RawADC = analogRead(temperaturePin);
  long Resistance;
  double Temp;
  // Assuming a 10k Thermistor. Calculation is actually: Resistance =
  (1024/ADC)
  Resistance=((10240000/RawADC) - 10000);
  // Utilizes the Steinhart-Hart Thermistor Equation:
  // Temperature in Kelvin = 1 / {A + B[ln(R)] + C[ln(R)]^3}
  // where A = 0.001129148, B = 0.000234125 and C = 8.76741E-08
  Temp = log(Resistance);
  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp *
Temp * Temp));
  Temp = Temp - 273.15; // Convert Kelvin to Celsius
  return Temp; // Return the Temperature
}
```

Serial Monitor (Figure 26-17):

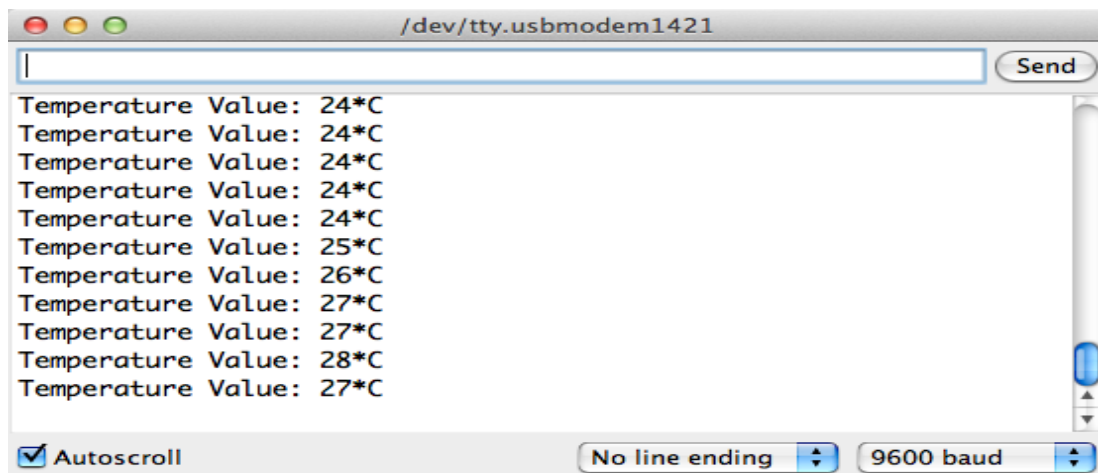


Figure 26-17: Serial Monitor output from the NTC Thermistor Example

26.5 Arduino Shields

Shields (Figure 26-18) are boards that can be plugged on top of the Arduino in order to extend its capabilities.

Some popular shields are Arduino Wi-Fi Shield, Arduino Ethernet shield and Arduino GSM Shield.

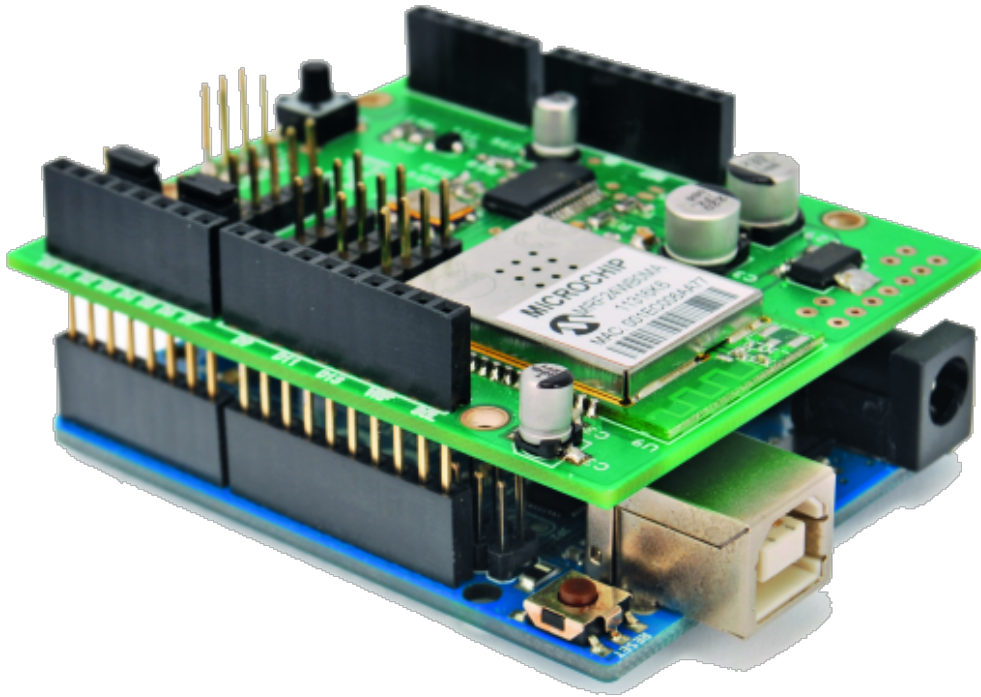


Figure 26-18: Arduino Shields

For more information about Arduino shields:

<https://www.arduino.cc/en/Main/ArduinoShields>

With the Arduino Wi-Fi Shield or Arduino Ethernet Shield you can connect an Arduino board to the internet. It can serve as either a server accepting incoming connections or a client making outgoing ones.

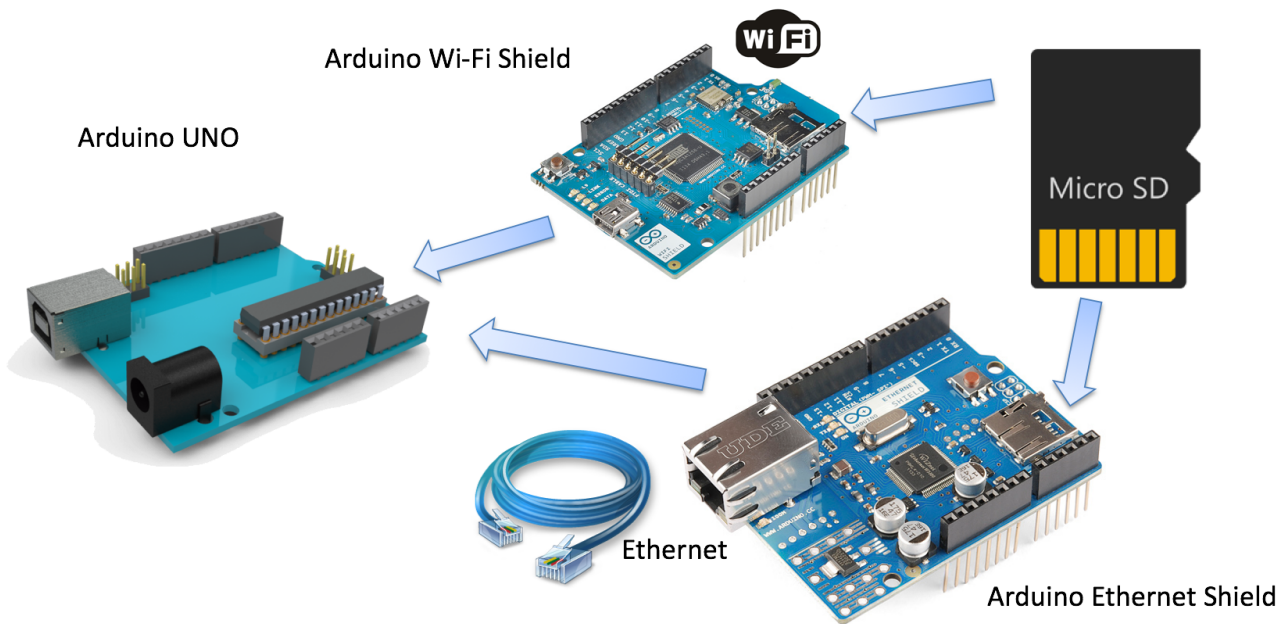


Figure 26-19: Arduino Wi-Fi/Ethernet Shields

Arduino Ethernet Shield

<http://arduino.cc/en/Reference/Ethernet>

Arduino Wi-Fi Shield

<http://arduino.cc/en/Reference/WiFi>

Both the Arduino Ethernet Shield and the Arduino Wi-Fi Shield has an SD card reader. See the SD Library: <http://arduino.cc/en/Reference/SD>

26.6 XBee

What is XBee? XBee are a popular wireless transceivers for a number of reasons.

They're flexible – they send and receive data over a serial port, which means they're compatible with both computers and microcontrollers (like Arduino).

They are highly configurable – you can have meshed networks with dozens of XBees, or just a pair swapping data.

You can use them to remotely control your robot, or arrange them all over your house to monitor temperatures or lighting conditions in every room.

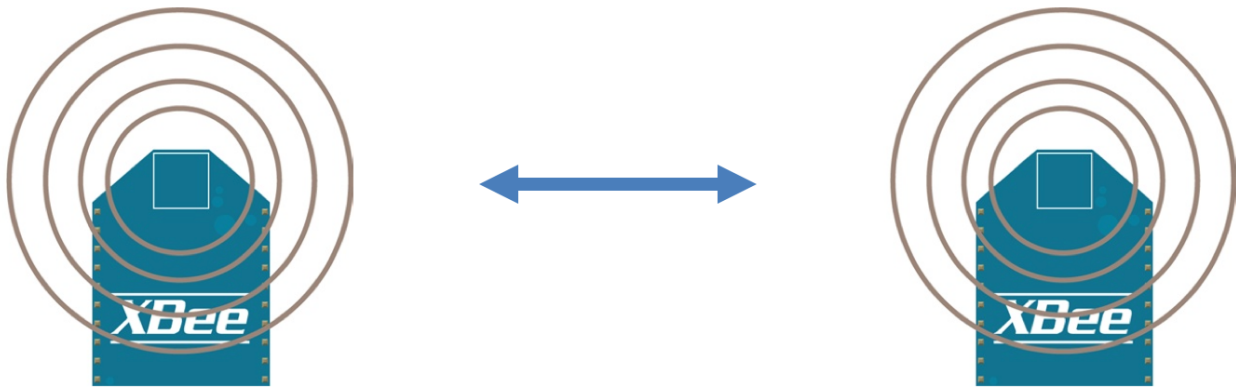


Figure 26-20: XBee Wireless Communication

<https://en.wikipedia.org/wiki/XBee>

XBee is the brand name from Digi International for a family of form factor compatible radio modules.

www.digi.com/xbee/

XBees are tiny blue chips that can communicate wirelessly with each other

These modules use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking.

XBee uses the ZigBee standard and adds to it and wraps it up in their own neat little package.

<http://www.zigbee.org/>.

Most of the Xbee modules operate at 2.4GHz

See also: https://www.sparkfun.com/pages/xbee_guide

26.7 XBee Hardware

You need 2 things:

1. XBee Modules, lots of different types exist
2. XBee Adapter Boards - Examples:
 - XBee USB Adapter
 - XBee SIP Adapter
 - XBee 5V/3.3V Adapter

- Arduino Wireless SD Shield (with XBee connector)

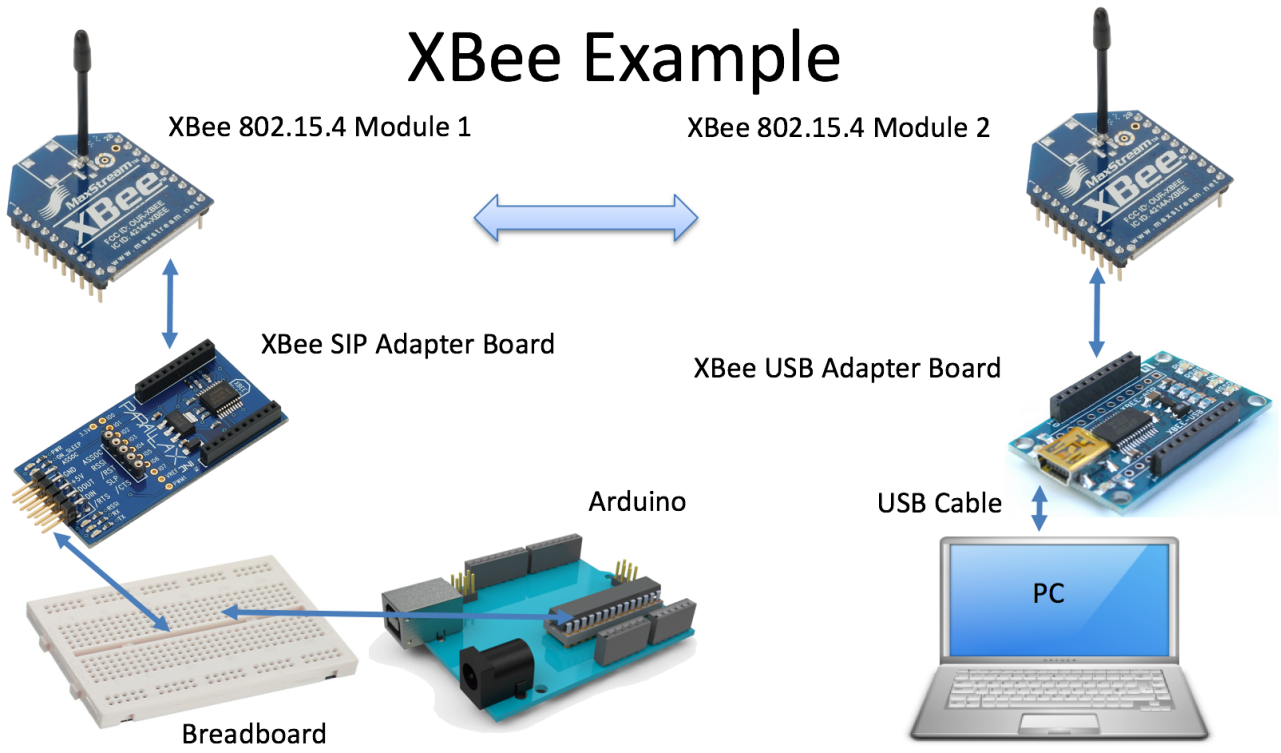
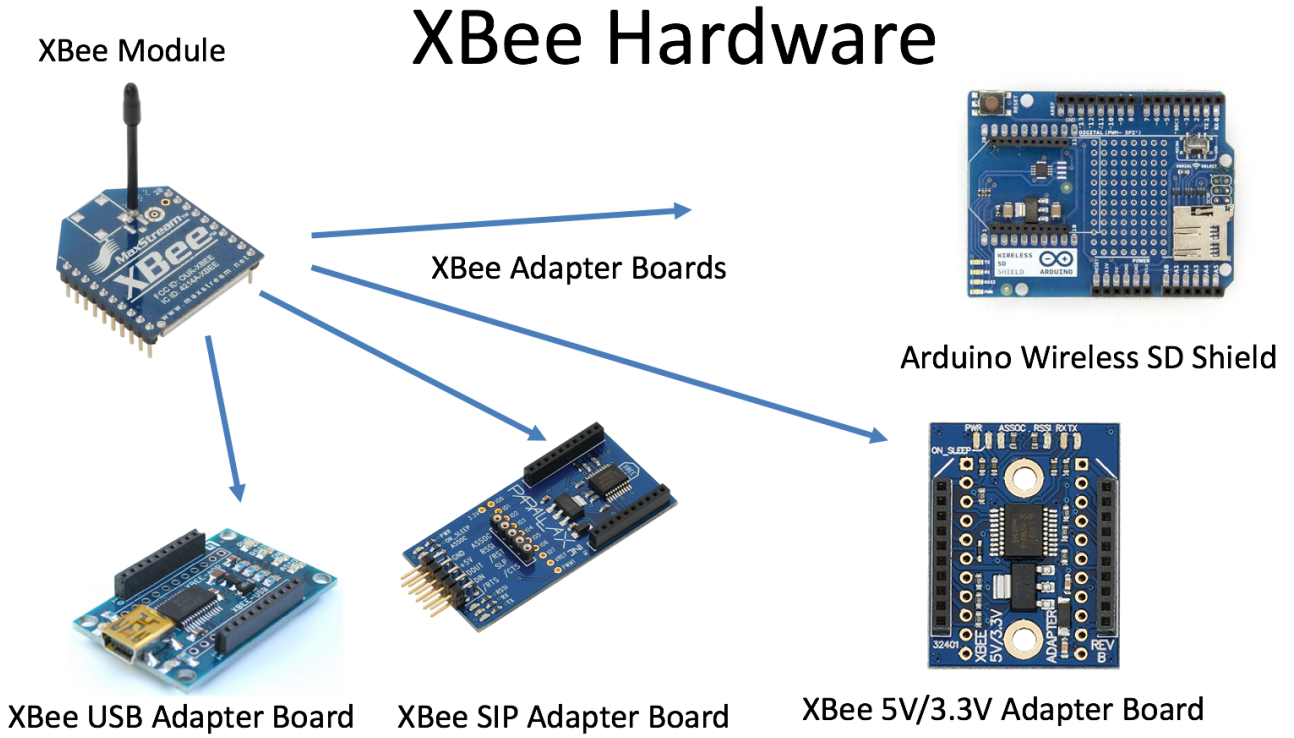


Figure 26-21: XBee Communication Example

26.8 Fritzing

An open source tool for making simple wiring diagram for your hardware wiring.

<https://en.wikipedia.org/wiki/Fritzing>

<http://fritzing.org>

27 Raspberry Pi

Web: https://www.halvorsen.blog/documents/technology/iot/raspberry_pi.php

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. The Raspberry Pi can run Linux and Windows 10 IoT Core.

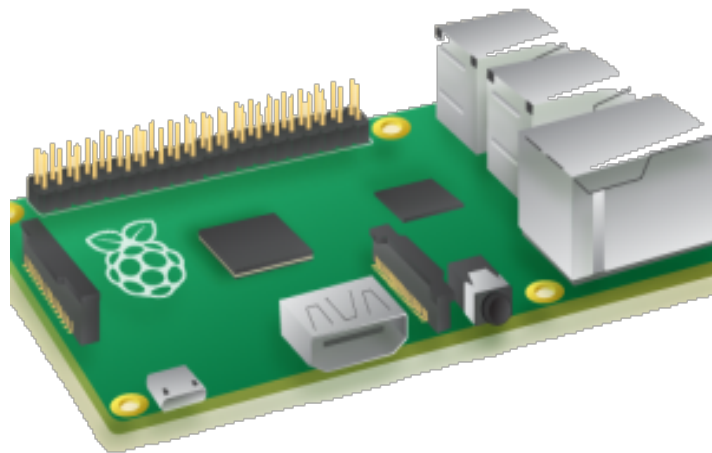
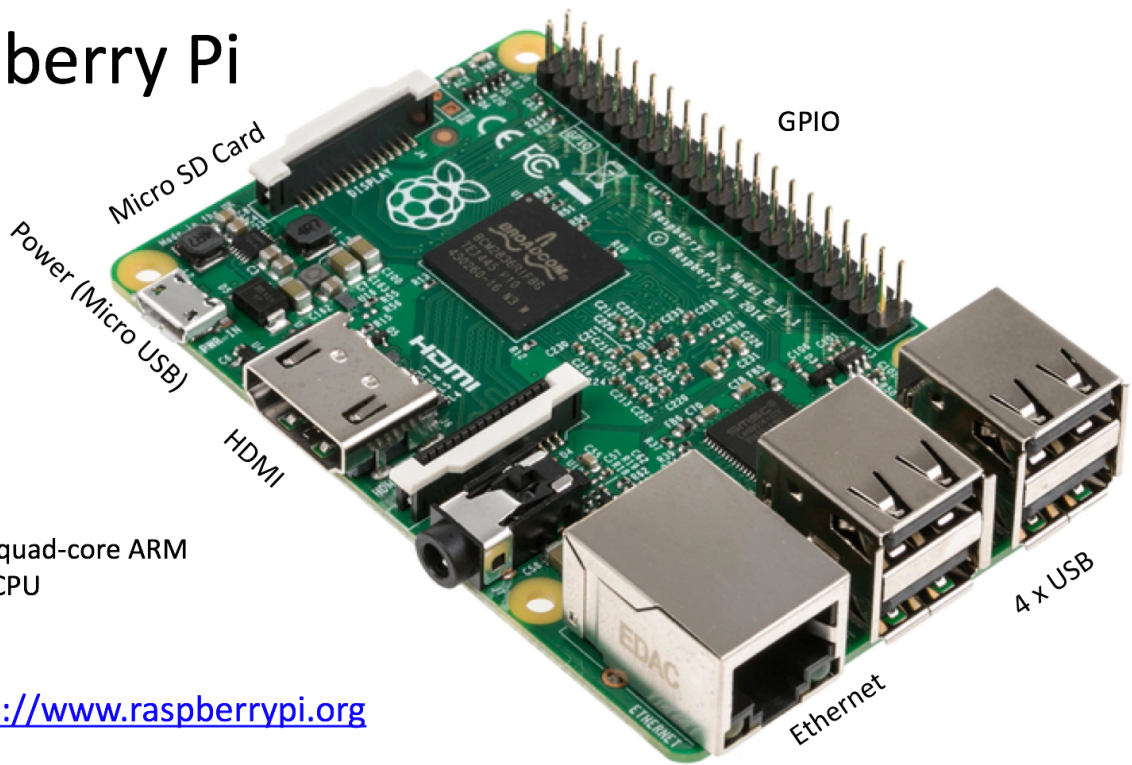


Figure 27-1: Raspberry Pi

The Raspberry Pi has the following specifications:

- A 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM
- Micro SD Card
- Power (Micro USB)
- 3.5mm audio jack/composite video
- Wi-Fi and Bluetooth
- 40 GPIO pins
- 4 x USB 2.0
- Ethernet
- 13x - GPIO pins
- 2x - SPI buses
- 1x - I2C bus
- 2x - 5V power pins
- 2x - 3.3V power pins
- 8x - Ground pins

Raspberry Pi



A 900MHz quad-core ARM Cortex-A7 CPU
1GB RAM

<http://www.raspberrypi.org>

Figure 27-2: Raspberry Pi

Raspberry Pi 2 - Overview

The Raspberry Pi 2 is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. The Raspberry Pi 2 can run Windows 10 IoT Core.

A 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM

40 GPIO pins

3.5mm audio jack/composite video

Small-Scale Computer

Raspberry Pi

Windows 10
Windows 10 IoT Core

- 13x - GPIO pins
- 2x - SPI buses
- 1x - I2C bus
- 2x - 5V power pins
- 2x - 3.3V power pins
- 8x - Ground pins

Figure 27-3: Raspberry Pi Hardware and Connectors

The Raspberry Pi 2 type B runs a quad-core ARM Cortex-A7 CPU and 1 GB RAM. It offers the following Connectors:

- 4 x USB 2.0 sockets
- 10/100 BaseT Ethernet socket
- HDMI video socket
- RCA composite video socket
- microSD card socket
- Powered from microUSB socket
- 3.5 mm audio out jack
- Header for GPIO and serial buses (I2C and SPI)
- Display Serial Interface (DSI) 15-way flat flex cable connector with two data lanes and a clock lane
- Camera connector 15-pin MIPI Camera Serial Interface (CSI-2)

Figure 27-4 shows the Raspberry Pi Pin Mappings:

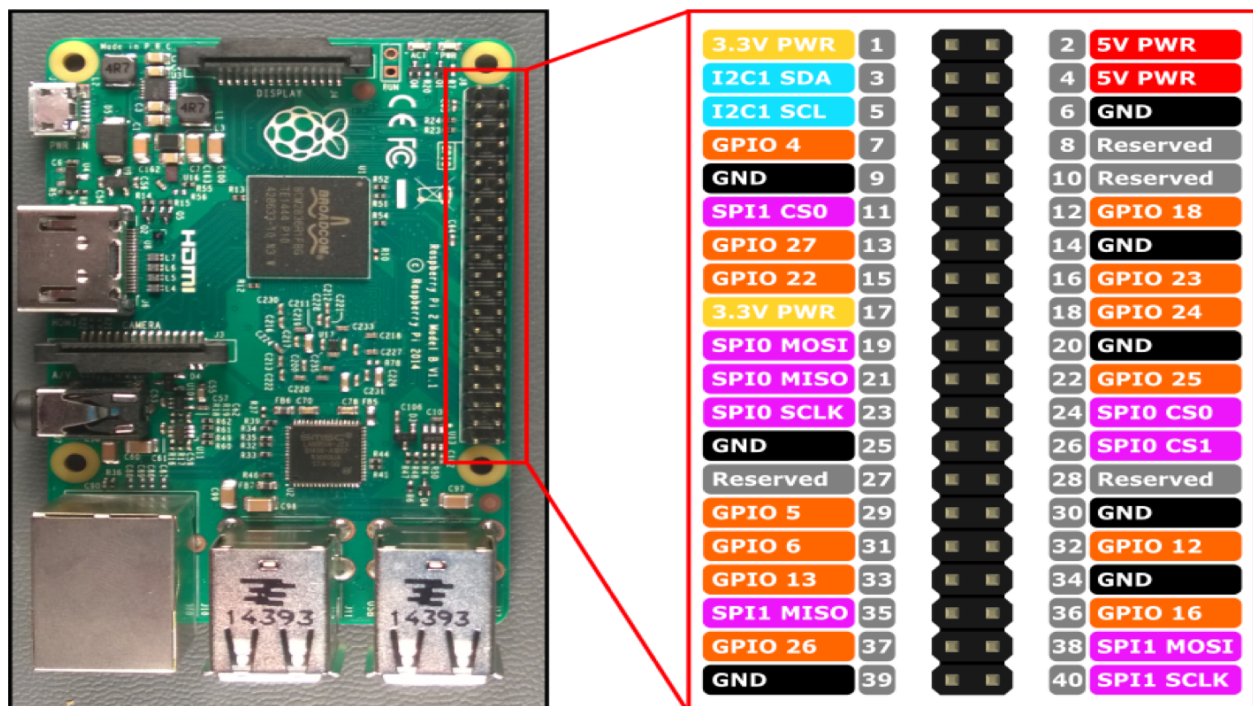


Figure 27-4: Raspberry Pi – Pin Mappings

27.1 Accessories

Figure 27-5 shows some official accessories that can be used with Raspberry Pi.

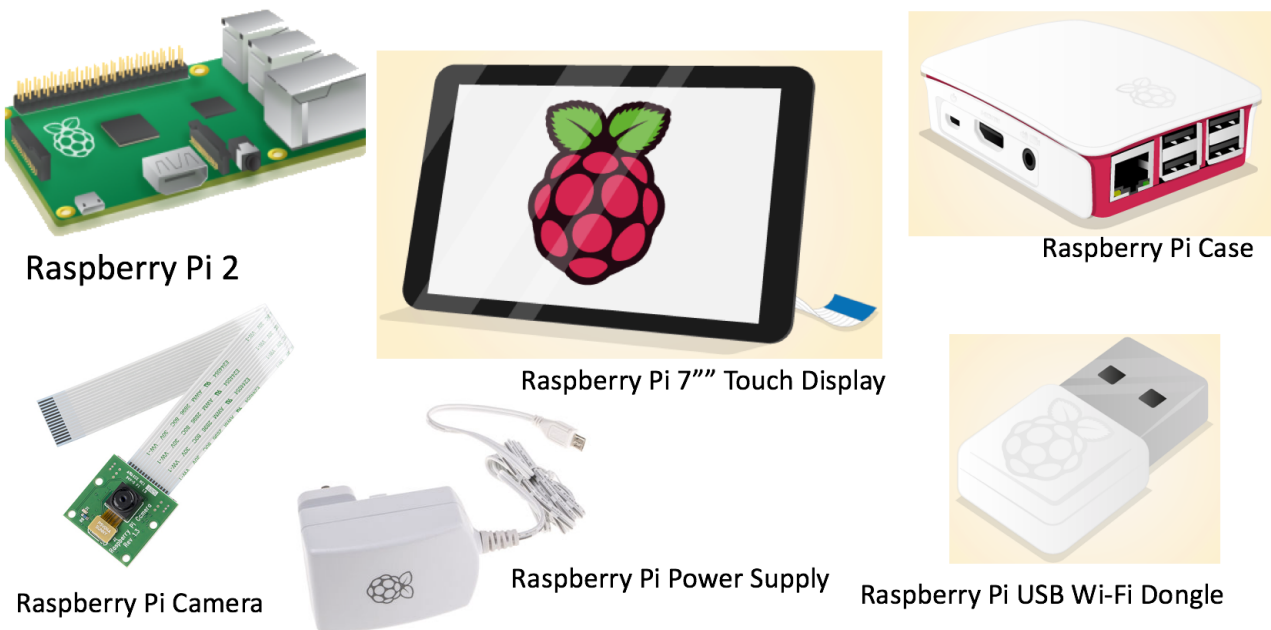


Figure 27-5: Raspberry Pi Accessories

27.2 Communication Protocols

Raspberry Pi supports the following protocols:

- UART (Universal Asynchronous Receiver/Transmitter,)– http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
- SPI (Serial Peripheral Interface)– http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
- I2C (Inter-Integrated Circuit)– <http://en.wikipedia.org/wiki/I2C>

27.3 Windows 10 IoT Core

Windows 10 IoT Core is a small scaled version of Windows running on small devices such as Raspberry Pi 2.

More information: <https://dev.windows.com/iot>

28 Industry 4.0

Web: <https://www.halvorsen.blog/documents/technology/industry40/>

The new “buzzword” for the combination of industry and the current Internet of Things (IoT) technology is Industry 4.0.

The term was first used in 2011 in Germany. In October 2012, a Working Group on Industry 4.0 presented a set of Industry 4.0 implementation recommendations to the German federal government. The high-tech strategy document outlined a plan to almost fully computerize the manufacturing industry without the need for human involvement[3].

So, is it just a “buzzword” or is it just “same shit – new wrapping”?

First things first – this isn't a new technology. Nor is it a business discipline. It is in fact a new approach to achieve results that weren't possible 10 years ago thanks to advancements in technology over the past decade.

Reference:

<http://www.techradar.com/news/world-of-tech/future-tech/5-things-you-should-know-about-industry-4-0-1289534>

Industry 4.0 is called the fourth industrial revolution (see Figure 28-1).

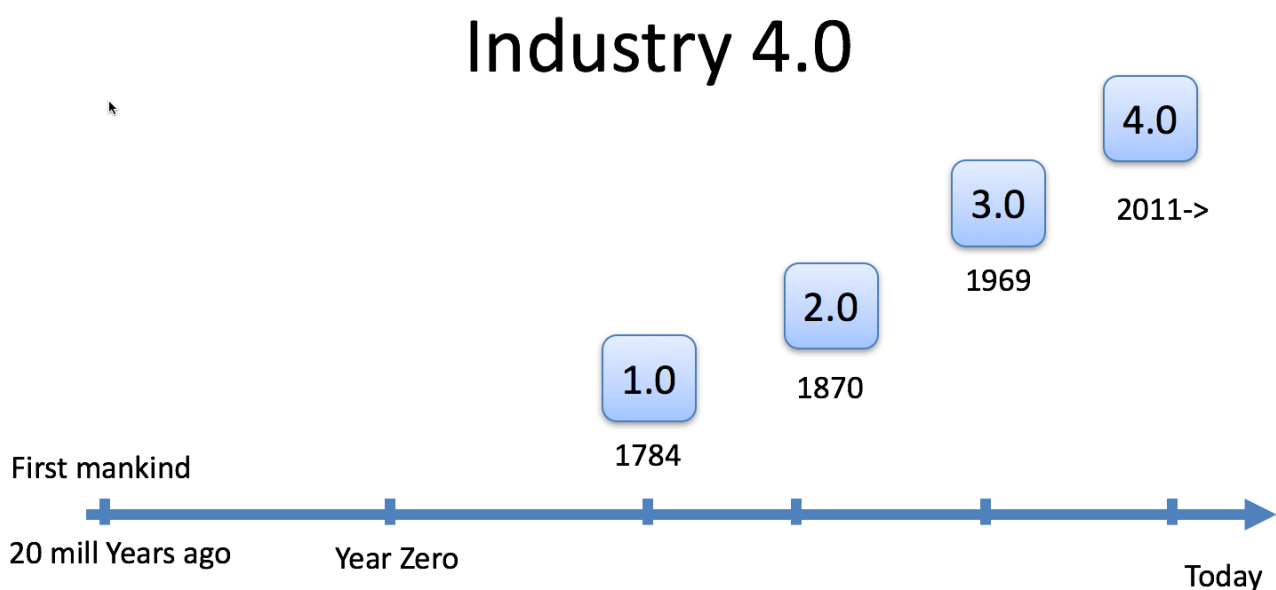


Figure 28-1: The Beginning of Industry 4.0

The first Industrial Revolution used water and steam power to mechanize production. The second used electric power to create mass production. The third used electronics and information technology to automate production. Now a fourth Industrial Revolution is building on the third, the digital revolution that has been occurring since the middle of the last century. It is characterized by a fusion of technologies that is blurring the lines between the physical, digital, and biological spheres.

Reference:

<http://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond>

Figure 28-2 shows some important steps in the Industrial IT and Automation evolution and the beginning of Industry 4.0.

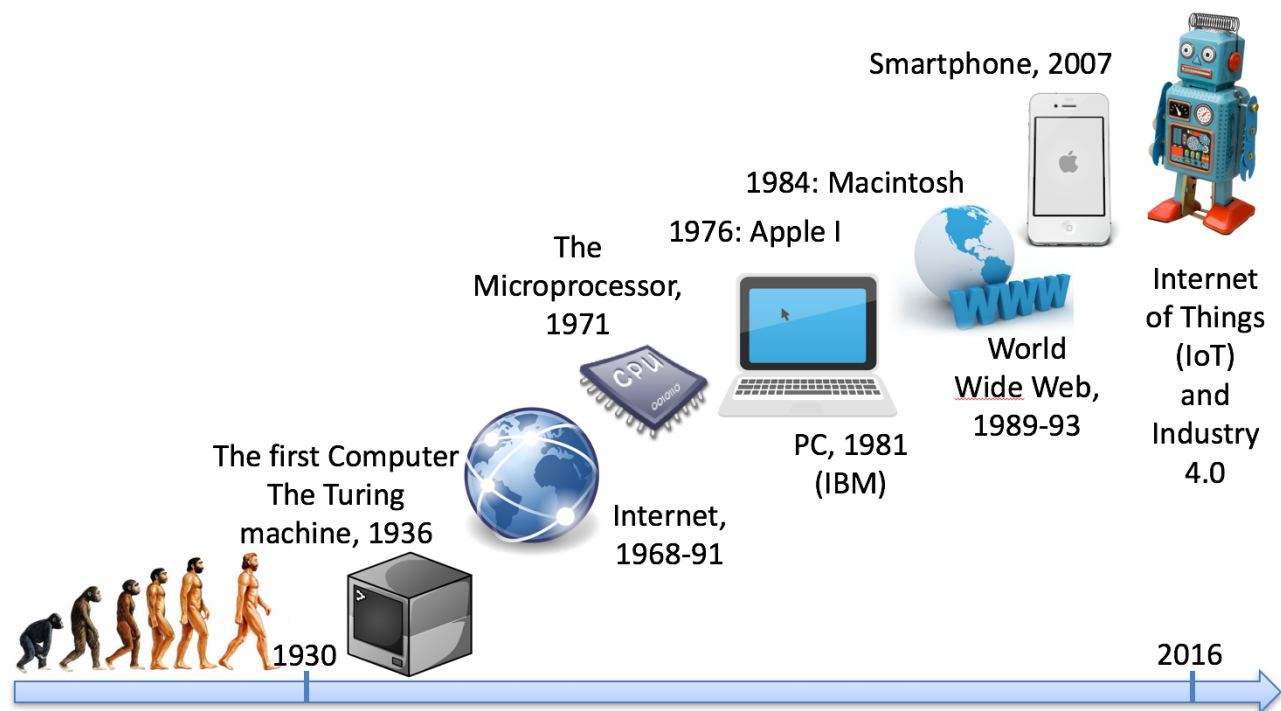


Figure 28-2: Industry 4.0 – The Fourth Industrial Revolution?

29 Machine Learning

Web: https://www.halvorsen.blog/documents/technology/machine_learning/

Machine Learning is all about Data Analytics, complex Mathematical Models and Algorithms, used for Predictive Analytics. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization.

- Machine Learning is about examine large amount of data (“Big Data”) looking for Patterns.
- It applies statistical techniques to large amounts of data, looking for the best pattern to solve your problem. This pattern can be referred to as a data model.
- The machine learning process starts with raw data and ends up with a model derived from that data.
- The machine learning algorithm is run on prepared data, and the result is referred to as a model.
- The knowledge gained is then used for Predictions, i.e., Predict the Future Machine Learning is an iterative process, which continuously updates the model when new data/knowledge arrives.

Machine Learning and Artificial Intelligence (AI) is closely related. You could say that Machine Learning is a tool to achieve Artificial Intelligence (AI).

Another term used a lot lately is Deep Learning, which is also closely related to Machine Learning and Artificial Intelligence. You could say Deep Learning is a technique for implementing Machine Learning. See Nvidia (2017) for more details about these differences.

Figure 29-1 shows a simplified sketch of the Machine Learning process:

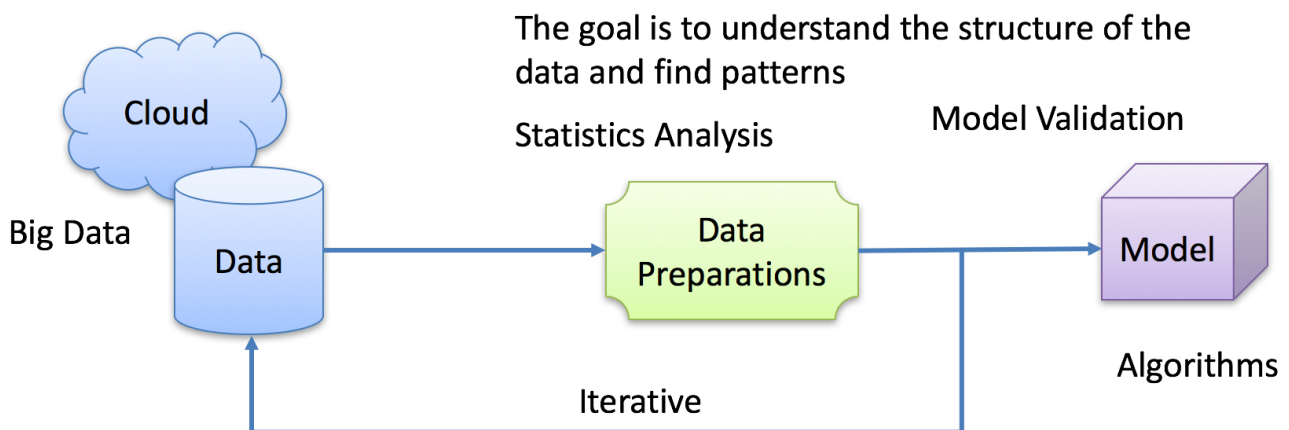


Figure 29-1: Machine learning

Machine Learning is about examine large amount of data (“Big Data”) looking for Patterns.

- It applies statistical techniques to large amounts of data, looking for the best pattern to solve your problem. This pattern can be referred to as a data model.
- The machine learning process starts with raw data and ends up with a model derived from that data.
- The machine learning algorithm is run on prepared data, and the result is referred to as a model.
- The knowledge gained is then used for Predictions, i.e., Predict the Future
- Machine Learning is an iterative process, which continuously updates the model when new data/knowledge arrives.

Machine Learning Applications and Examples:

- Create complex Weather models from a large amount of collected weather data. The weather models are then used to predict the weather in the future (short or long termed)
- Transportation: Self driving cars, ships or so-called Autonomous vehicles
- Marketing and sales, e.g., Online recommendation offers such as those from Amazon and Netflix
- Apple Siri (intelligent personal assistant) and similar services
- Financial services, such as Stock market, etc.
- ... hundreds of other examples

For more information about Machine Learning:

https://halvorsen.blog/documents/technology/machine_learning/

29.1 Machine Learning in Automation Systems

In Automation Systems, more traditional and well known “Machine Learning” principles such as System Identification, State Estimation with Kalman Filter and Model Predictive Control (MPC) are used. Machine Learning is all about Data Analytics, complex Mathematical Models and Algorithms used for Predictive Analytics.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization. In System Identification, State Estimation (Kalman Filter) and Model Predictive Control (MPC) all these things apply.

29.1.1 System identification

System Identification uses statistical methods to build mathematical models of dynamical systems from measured data.

We have 2 main categories of System Identification:

- **Parameter Estimation** based on that we have developed a mathematical model using the laws of physics (Mechanistic Models) and you want to find the unknown model parameters. Here we will use least squares method as an example. The unknown parameters are then found from experimental data.
- **Black-box / Subspace methods:** System Identification based on that you do not have a mathematical model available. The models (Empirical Models) are found from experimental data only using advanced algorithms.

See Figure 29-2.

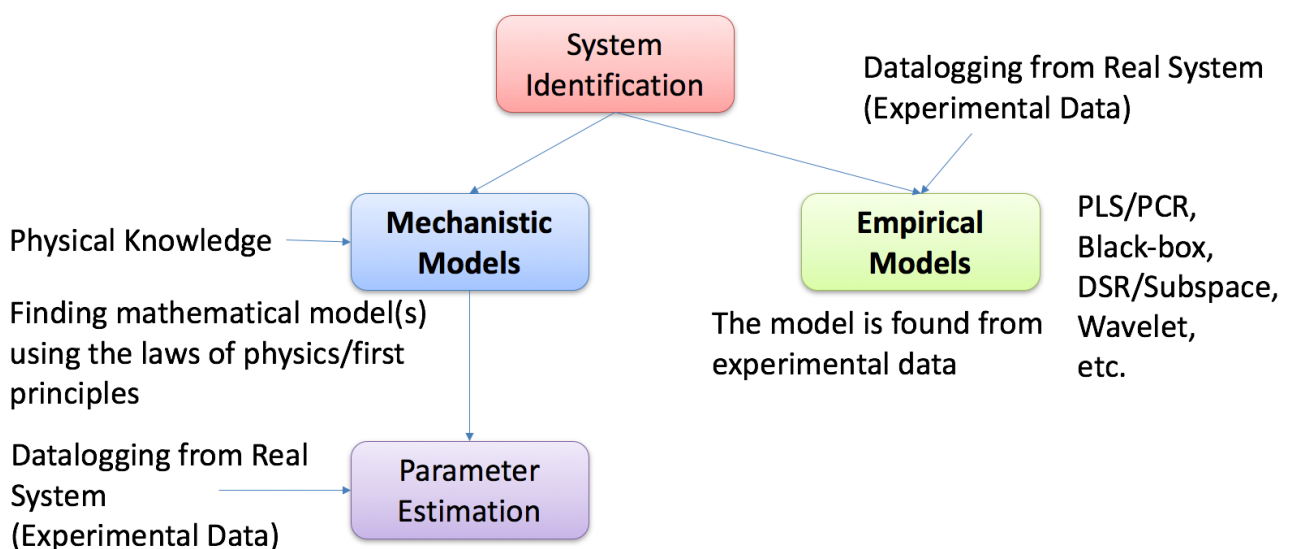


Figure 29-2: System Identification

For more information, see:

<https://www.halvorsen.blog/documents/automation/>

29.1.2 State Estimation

State Estimation uses mathematical models in order to estimate the internal states of a process

The Kalman Filter is a commonly used method to estimate the values of state variables of a dynamic system that is excited by stochastic (random) disturbances and stochastic (random) measurement noise.

LabVIEW has built-in functionality for both System Identification and State Estimation

For more information, see:

<https://www.halvorsen.blog/documents/automation/>

29.1.3 Model Predictive Control (MPC)

Model predictive control (MPC) is an advanced method of process control that has been in use in the process industries since the 1980s.

Model Predictive Control (MPC) is a multivariable control algorithm.

Model predictive controllers rely on dynamic models of the process, most often linear empirical models obtained by system identification.

MPC is based on iterative, finite-horizon optimization of a plant model.

This is achieved by optimizing a finite time-horizon, but only implementing the current timeslot. MPC has the ability to anticipate future events and can take control actions accordingly.

More information about MPC:

<https://www.halvorsen.blog/documents/automation/mpc/>

Part 5 : Applications and Examples

In this part, we will go through some examples where many of the topics in this document has been used.

30 Weather Station

At the university, we have established a Weather System for presenting the weather at the university.

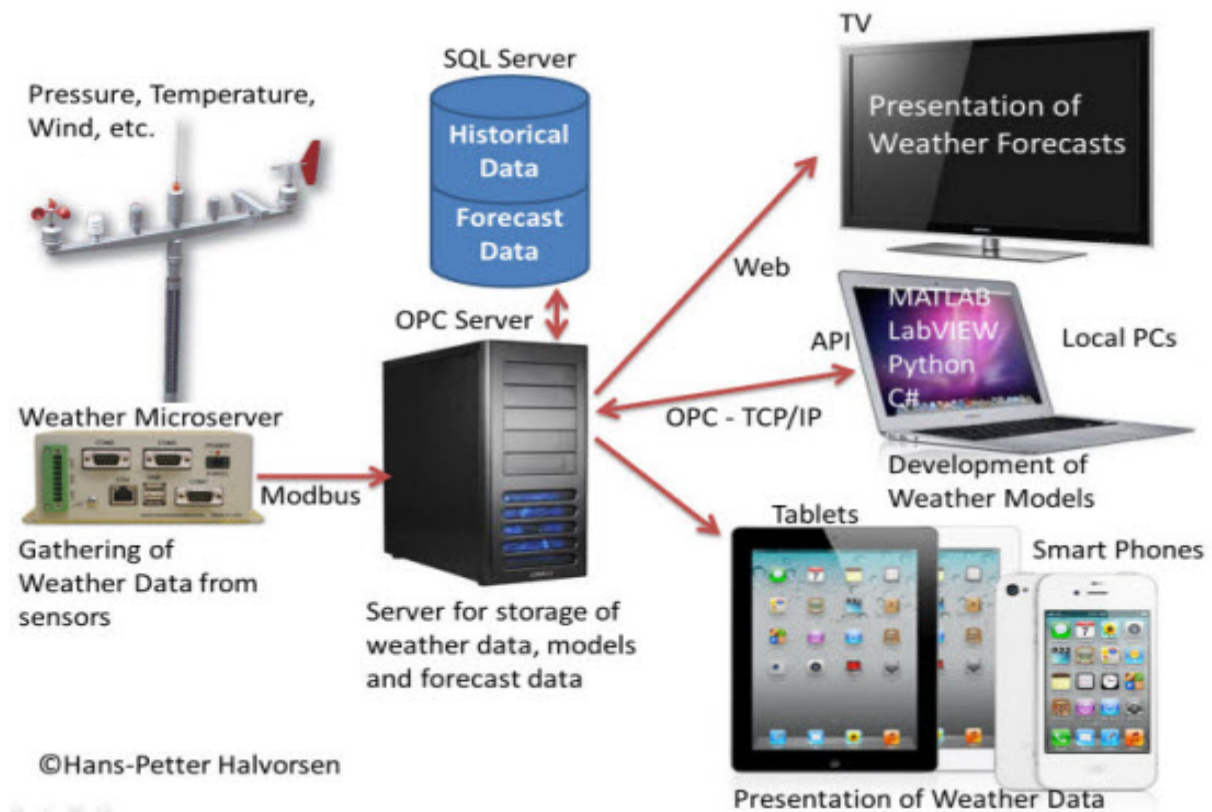


Figure 30-1: Weather Station Overview

The system is showing the weather data from the weather station located at the university. The system shall be used to create weather models and forecasting. The weather system has a SDK/API that makes it possible to retrieve data in different manners, including Web Services. APIs have also been created for the following languages: C#, LabVIEW, MATLAB and Python. An OPC API is also available in these 4 languages. These APIs make it possible to retrieve data from the system and create your own weather models used in forecasting and weather prediction.

In the future, the weather system will be extended with more features, including more advanced weather prediction and forecasts, but also native Apps for iOS, Android and Windows 8 will be developed.

A Web site is part of the solution (Figure 30-2).

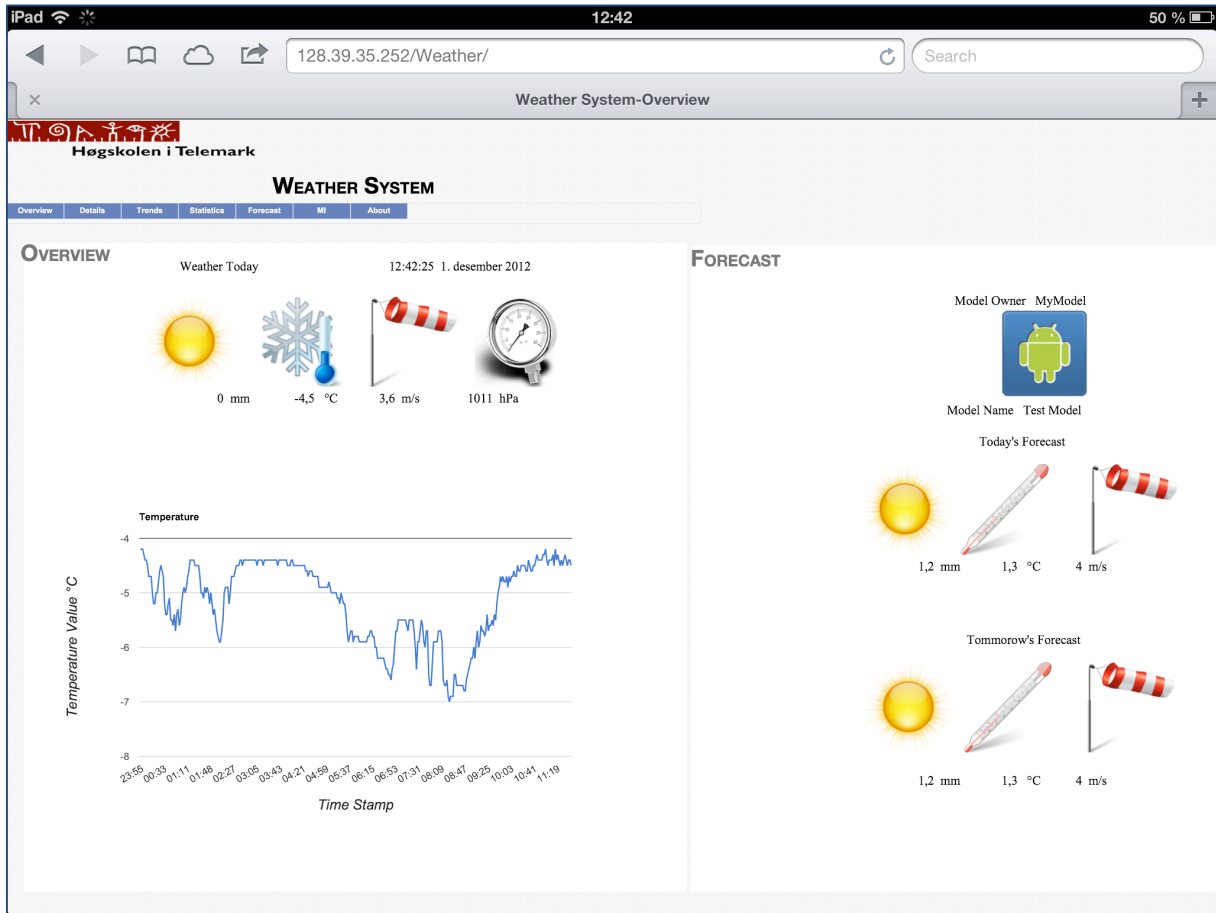


Figure 30-2: Weather Station Web Site

The system consists of the following modules:

Name	Description	Implementation
Modbus Data Service	Get Weather Data into the Database	LabVIEW
Weather Data Web Service	Publish Methods and Data	C#
Web Site	Show Weather Data on TV and in Web Browser	ASP.NET, C#
Tablet Data Web Service	Publish selected Weather Data to make it available on the Tablet	LabVIEW
Tablet "App"	Present Weather Data on a Tablet	LabVIEW + Data Dashboard for LabVIEW
Weather System SDK	Makes it possible to retrieve Weather Data, Create Models, an Weather Forecast	C#, LabVIEW, MATLAB, Python
Configuration Tool	Administrator Tool	LabVIEW or C#/WPF
Model Prediction Service	Execute Models and updates the Database with Forecast Data	LabVIEW
Native Tablet App	Native Tablet App(s) showing Weather Data and Forecast	iOS, Windows 8, Android

30.1 Database

In Figure 30-3 we see the ER diagram of the database used in the Weather System.

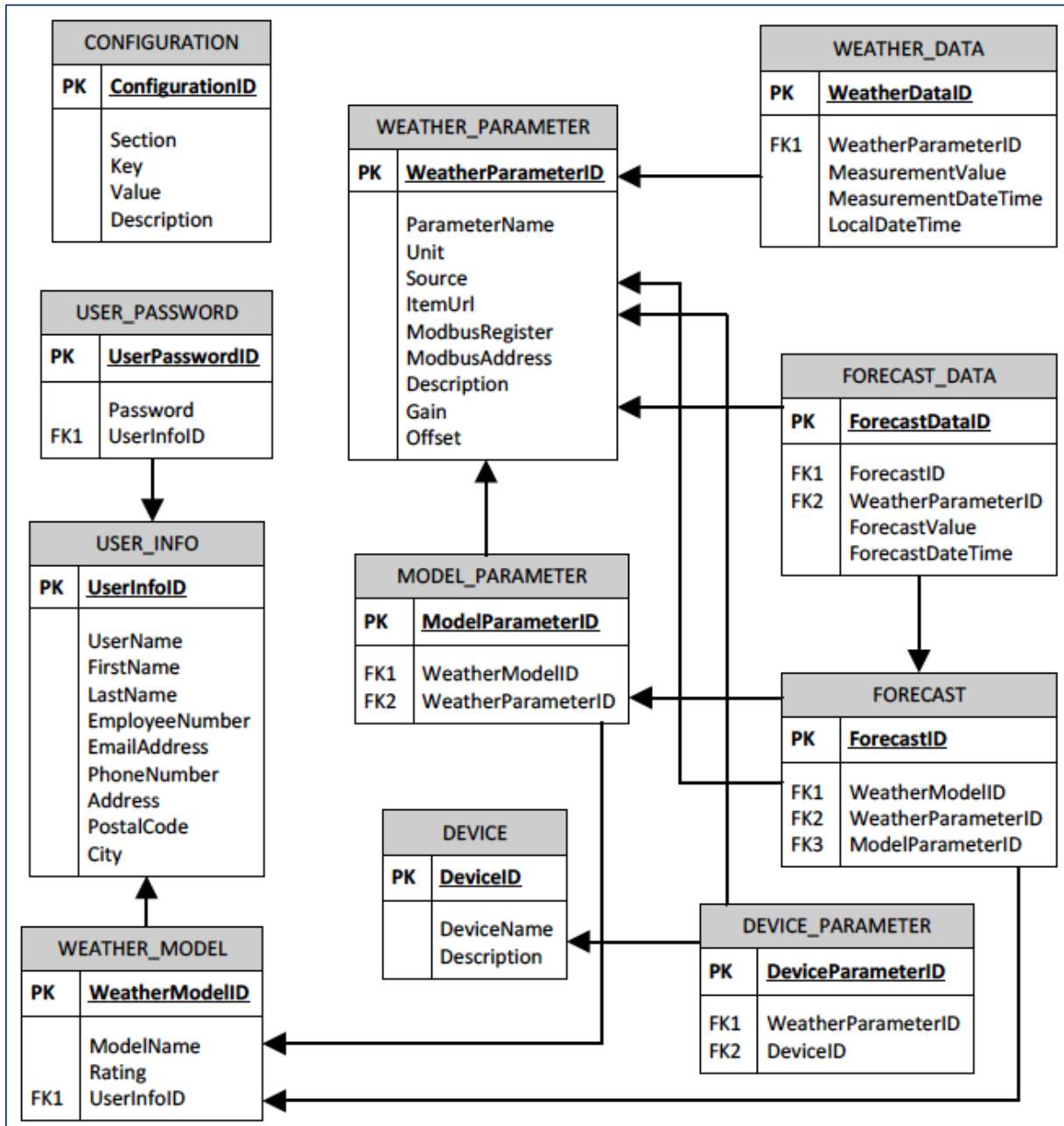


Figure 30-3: Database Diagram

A short description of the different tables:

Table Name	Description
CONFIGURATION	Contain data used as configuration parameters and may be utilized by the SQL server or other external applications.
USER_INFO	Information of the Owners' of the prediction models.
USER_PASSWORD	Store password of owners' of the prediction models.
DEVICE	External devices which use the service of the database such as; Modbus, Tablet and website.
DEVICE_PARAMETER	The external devices are able to pre-select needed parameters.
WEATHER_PARAMETER	Store all the sensors from the micro server, there are 68 parameters, but not all of them is used
WEATHER_DATA	All the values each parameters are updated in this table every two minutes. The table will increase automatically by time.
WEATHER_MODEL	Contains weather prediction models.
MODEL_PARAMETER	Parameters of the model.
FORECAST	Containing, weatherModelID, weatherParameterID, modelParameter ID
FORECAST_DATA	Output values of the prediction models. The predicted values.

30.2 OPC Server

The KEPware OPC server is used. You need to use a Tunneller software in order to connect to the OPC Server.

There exists OPC APIs for the following languages:

- C#
- LabVIEW
- MATLAB
- Python

30.3 Web Service

A Web Service has been made in order to get data from the system (Figure 30-4).

Web Service: <http://128.39.35.252/WebApi/WebService.aspx>

Below we see the available web service methods:

WebService

The following operations are supported. For a formal definition, please review the [Service Description](#).

- **GetDailyAverage**
This function takes a parameter name as input and returns daily average for the past 30 days Example: GetDailyAverage('umtTemp1')
- **GetHourlyAverage**
This function takes a parameter name as input and returns hourly average for the past 24 hours Example: GetHourlyAverage('umtTemp1')
- **GetLatestData**
This function takes a parameter name as input and returns its latest recorded data Example: GetLatestData('umtTemp1')
- **GetMaxMin**
GetMaxMin(string parameterName,string period,string mode) This function takes a parameter name, period and mode(max or min) as input and returns the maximum or minimum recorded data of the selected parameter in the selected period. available periods are:1DayAgo,1WeekAgo,1MonthAgo,1YearAgo Example: GetMaxMin('umtTemp1','1DayAgo','min')
- **GetSelectedLatestData**
This function takes an array of parameter names as input and returns their latest recorded data Example: GetSelectedLatestData(arrayOfParameterNames)
- **GetSelectedMaxMinData**
GetSelectedMaxMinData(string[] parameterName,string period,string mode) This function takes an array of parameter names,period and mode(max or min) as input and returns the maximum or minimum values for the selected parameters in the selected period.available periods are: 1DayAgo,1WeekAgo,1MonthAgo,1YearAgo Example: GetSelectedLatestData('arrayOfParameterNames','1WeekAgo','min')
- **GetSelectedWeatherItemsData**
This function takes an array of parameter names and a period as input and returns their recorded data in the entered period. Example: GetWeatherItemData(arrayOfParameters,'2012-11-02','2012-11-03')
- **GetWeatherData**
This function takes a period as input and returns recorded data for all parameters in this period. the input period can be:1HourAgo,1DayAgo,1WeekAgo,1MonthAgo,1YearAgo Example: GetWeatherData('1WeekAgo')
- **GetWeatherItemData**
This function takes a parameter name and a period as input and returns the recorded data of this parameter in the entered period. Example: GetWeatherItemData('umtTemp1','2012-11-02','2012-11-03')
- **GetWeatherParameters**
This function returns all the data Inside WEATHER_PARAMETER table in the Weather station database

Figure 30-4: Weather Station Web Service

30.4 iPad App

Below we see the iPad App that has been made for the Weather Station.

The screenshot shows the iPad app interface for 'Telemark University College - Weather System'. The app displays several weather metrics in a clean, modern layout:

- Temperature:** -4,417 Deg.C
- Wind Speed:** 3,136 m/s
- Rainfall:** 0 mm
- Wind Direction:** A circular gauge showing a reading of 354,4 degrees.
- Barometric Pressure:** 1011 hPa
- Wind Chill:** -9,417 Deg.C
- Humidity:** 89 %

At the bottom, there is a red link that says 'Visit Weather Station Web Site'. A large rooster icon is positioned on the right side of the screen, with a yellow arrow pointing to the right.

Figure 30-5: Weather Station iPad App

30.5 Windows 10 Universal App

In Figure 30-6 we see the Windows 10 Universal App that has been made.

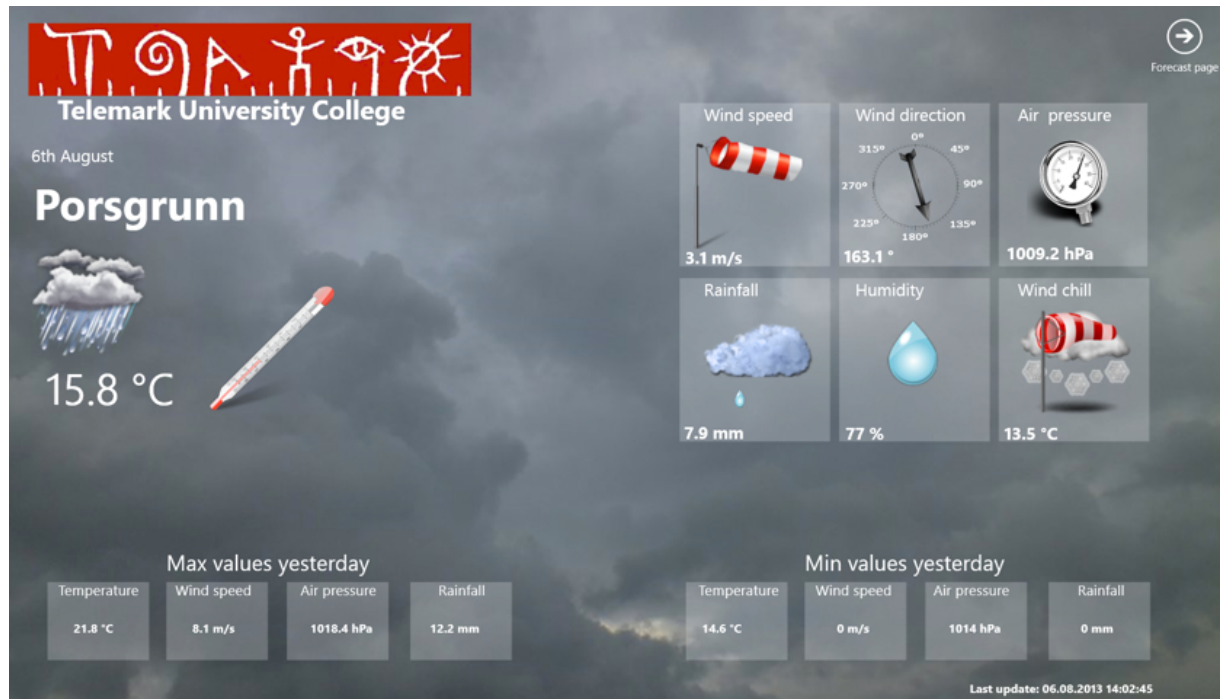


Figure 30-6: Weather Station Windows 10 Universal App

You can download the App from Windows Store for free.

31 DeltaV Training and Research Center

DeltaV is a Process Control System from Emerson, see Figure 31-1.

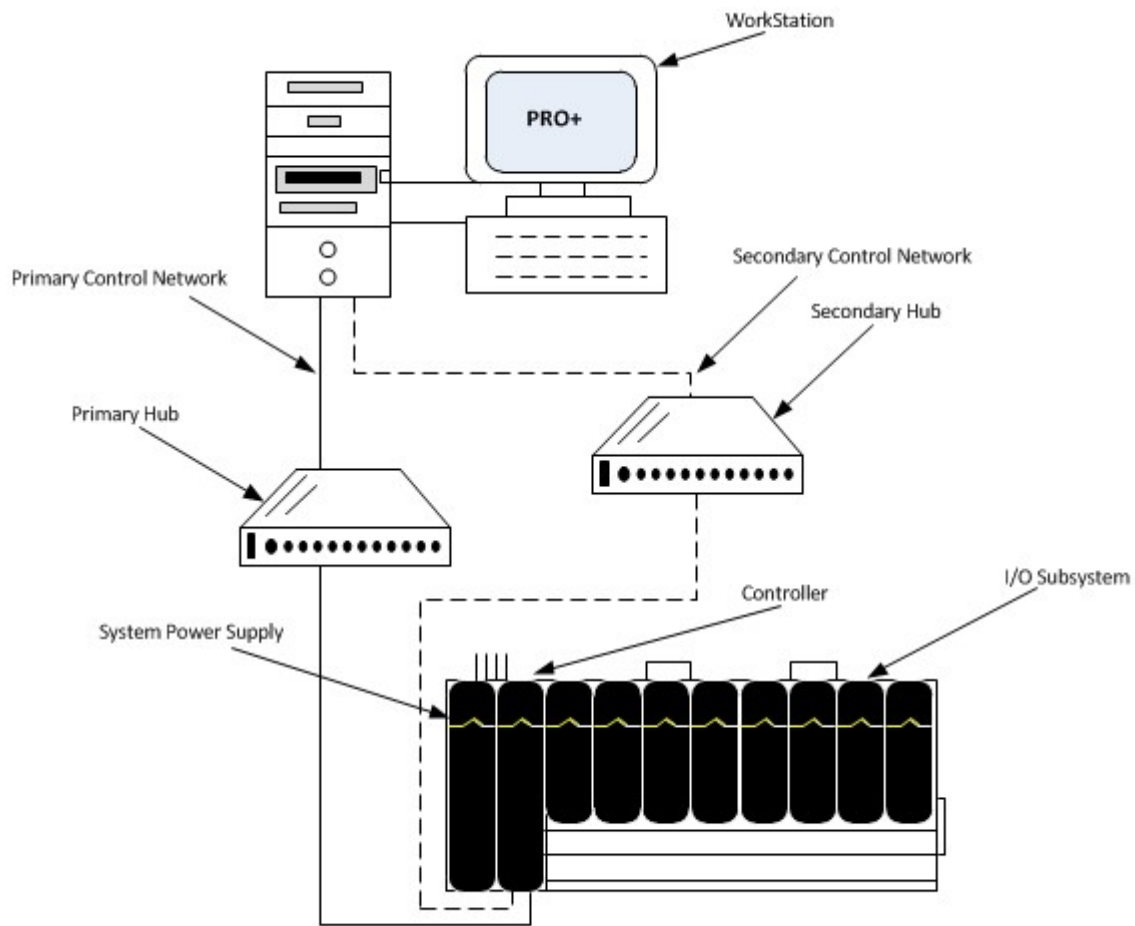


Figure 31-1: DeltaV Hardware Overview

Figure 31-2 shows the DeltaV Training and Research Center Web Portal that has been made.

Delta V



Research Center	Training Center
	
<p>Is a information portal for HiT's DeltaV distributed control system. On this website you will find:</p> <ul style="list-style-type: none"> • Basic informat ion about the DeltaV system • Basic information about the UBD Model and the Silo Model • A brief user manual • Specifications and schematics of the DeltaV system • Information about DeltaV student projects 	<p>Is a training portal for HiT's DeltaV Training Station On this website you will find:</p> <ul style="list-style-type: none"> • Basic information about DeltaV • Basic information about the DeltaV Training Station • Basic information about the Air heater, level control, two-tank, pt-100 • Video tutorials and written tutorials

Figure 31-2: DeltaV Training and Research Center Web Portal

You can access the Web Portal here: <http://128.39.35.248/DeltaV>

31.1 Training Center

An important part of the system is a Training Station (Figure 31-3) where students and staff (or others) can learn to use the DeltaV platform.

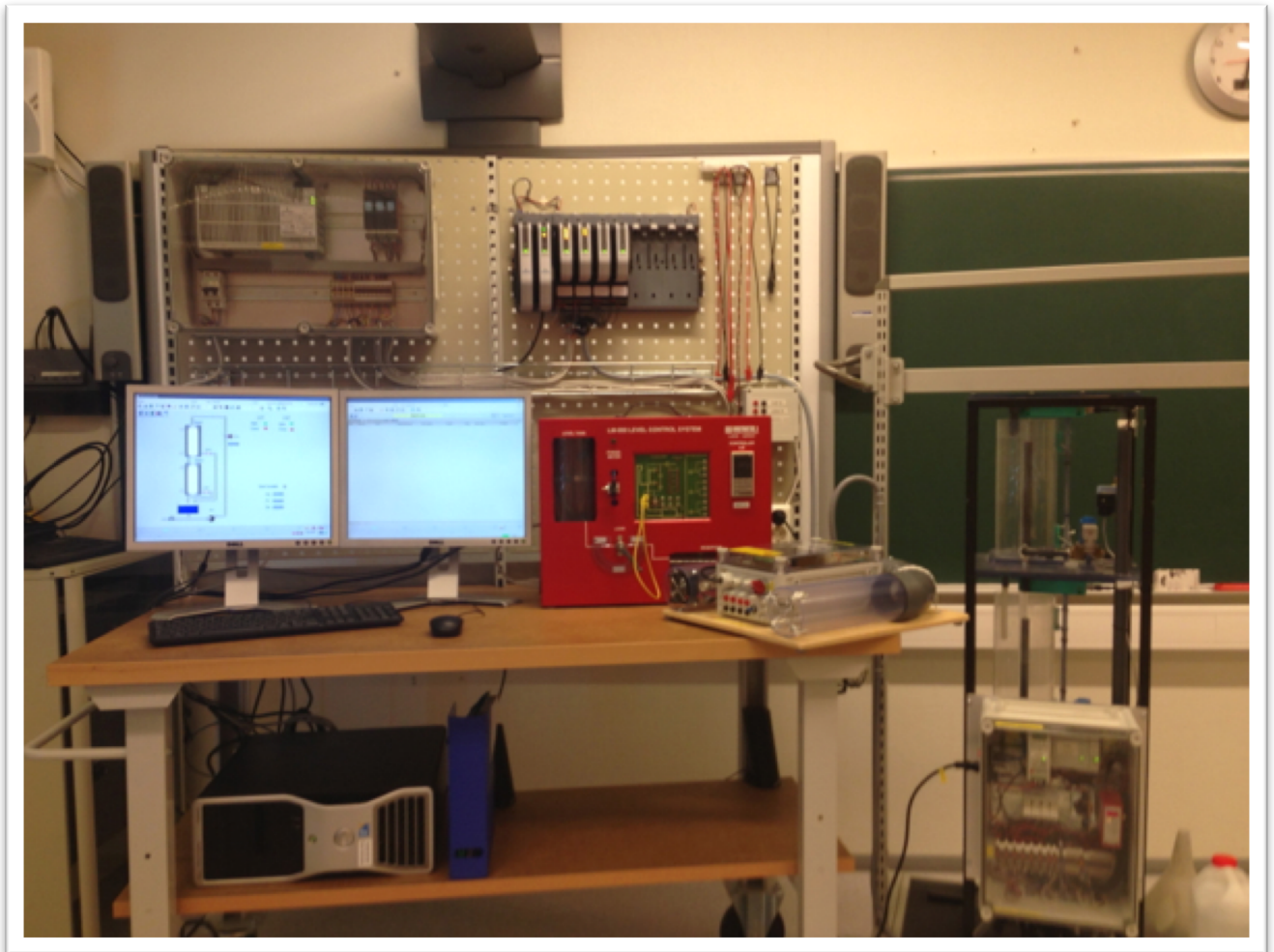
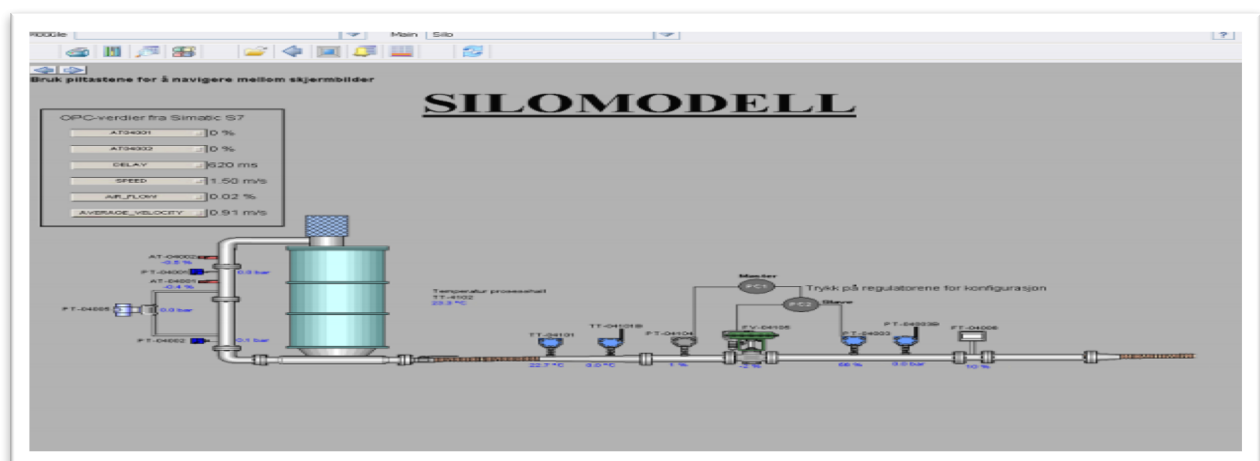


Figure 31-3: DeltaV Training Station

31.2 Research Center

The DeltaV Center has also several DeltaV facilities used in research.



Delta V

RESEARCH CENTER

Home

DeltaV

User Manual

Specifications

Projects

General Information

UBD Model

Silo Model

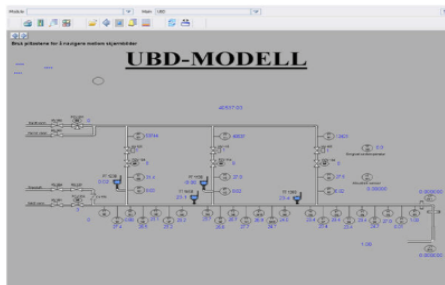
Underbalanced Drilling Model



UBD is short for "Underbalanced Drilling", an oil & gas drilling technique where the pressure in the wellbore is kept lower than the fluid pressure in the formation being drilled. This has several advantages.

[Watch Video - Basic principles of underbalanced drilling.](#)

The UBD model is a long pipe with various instruments. There's control valves, flow transmitters, wired/wireless pressure transmitters and wired/wireless temperature transmitters. In total there's 46 wired instruments connected to a data recording system from National Instruments, located in cabinet 4300. Further there is 4 wireless instruments. The wired data is sent from the NI unit to DeltaV over ethernet. The wireless data is sent to gateway with WirelessHART signals and then via Modbus to DeltaV.



UBD model screen in DeltaV Operate Run



Various types of instruments on UBD-model like valves, wired/wireless pressure transmitter and flow transmitter

All the work at the research facility are documented in the DeltaV Research Center Web Portal.

32 Data Management and Monitoring Platform

Figure 32-1 shows an overview of the Data Management and Monitoring Platform (DMM) that has been developed.

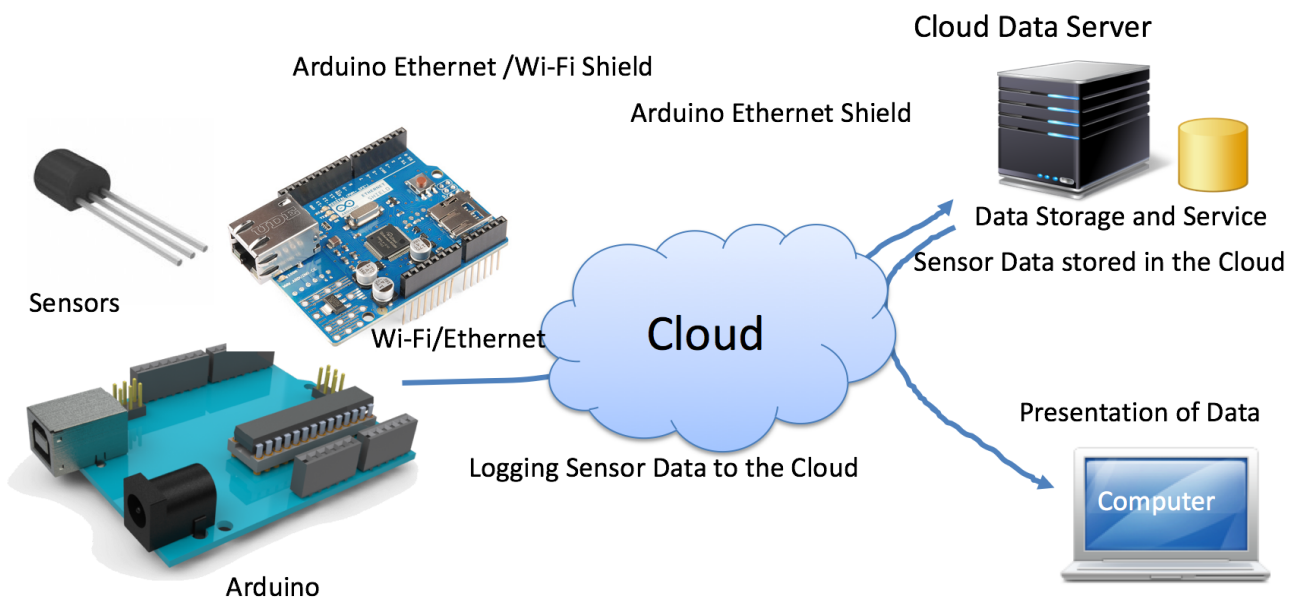


Figure 32-1: Overview of the Data Management and Monitoring Platform (DMM)

Database:

An important part of the DMM platform is the Database where all the configuration data, sensor data, etc. are stored (Figure 32-2).

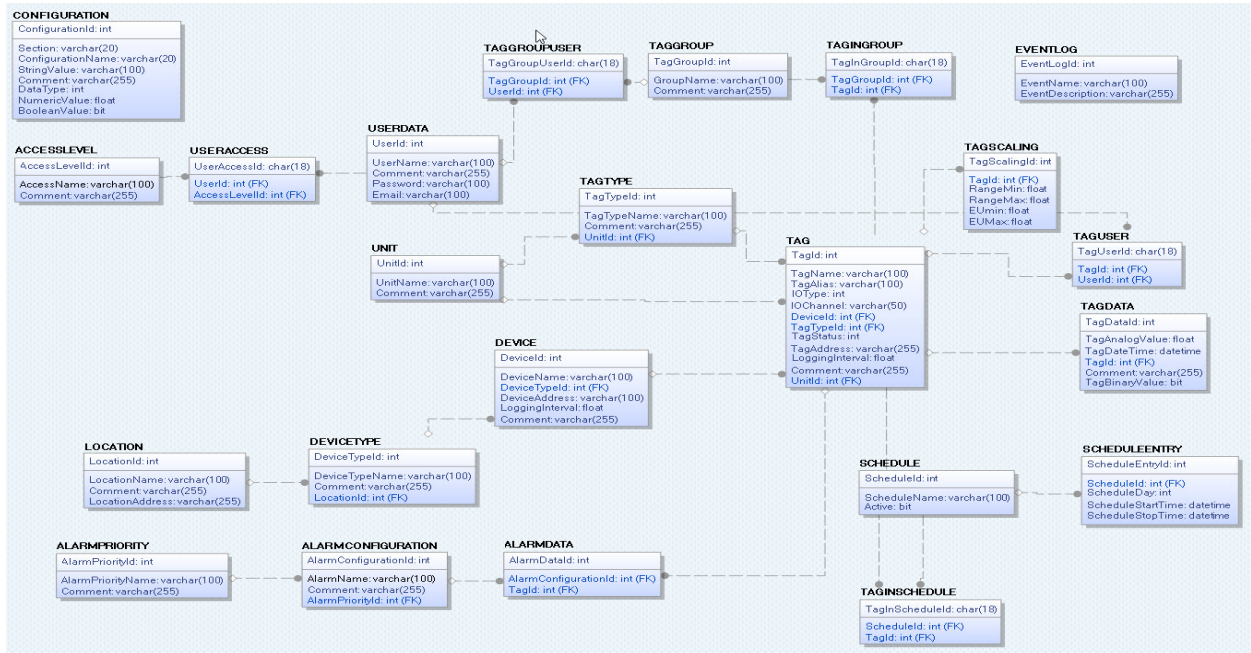


Figure 32-2: DMM Database

Management:

Here we can configure Devices and Tags (Figure 32-3, Figure 32-4)

Blog

Datalogging, Management and Monitoring

Datalogging, Management and Monitoring

Hans-Petter Halvorsen

Management and Configuration

Devices are typically sensor nodes that include one or more measurements, so-called tags.

- [Device Management](#)
- [Tag Management](#) - Create Tags and link them to a specific Device.

Monitoring

- [Chart](#)
- [Search Data](#)

Figure 32-3: DMM Platform - Management

Tags

Devices are typically sensor nodes that include one or more measurements, so-called tags. A Device can be based on hardware like Arduino, Raspberry Pi, and different DAQ devices/I/O modules.

Select Device:

List of Tags registered in the database:

TagId	Tag Name	Tag Alias	Action
8	Air Quality1	Air Quality Indoor	<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="View Data"/>
9	Humidity1	Humidity Indoor	<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="View Data"/>
1	Temperature1	Temperature Indoor	<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="View Data"/>

Figure 32-4: DMM Management – Tag Configuration

Monitoring:

Here (Figure 32-5) we can see the logged data with charting possibilities, etc.

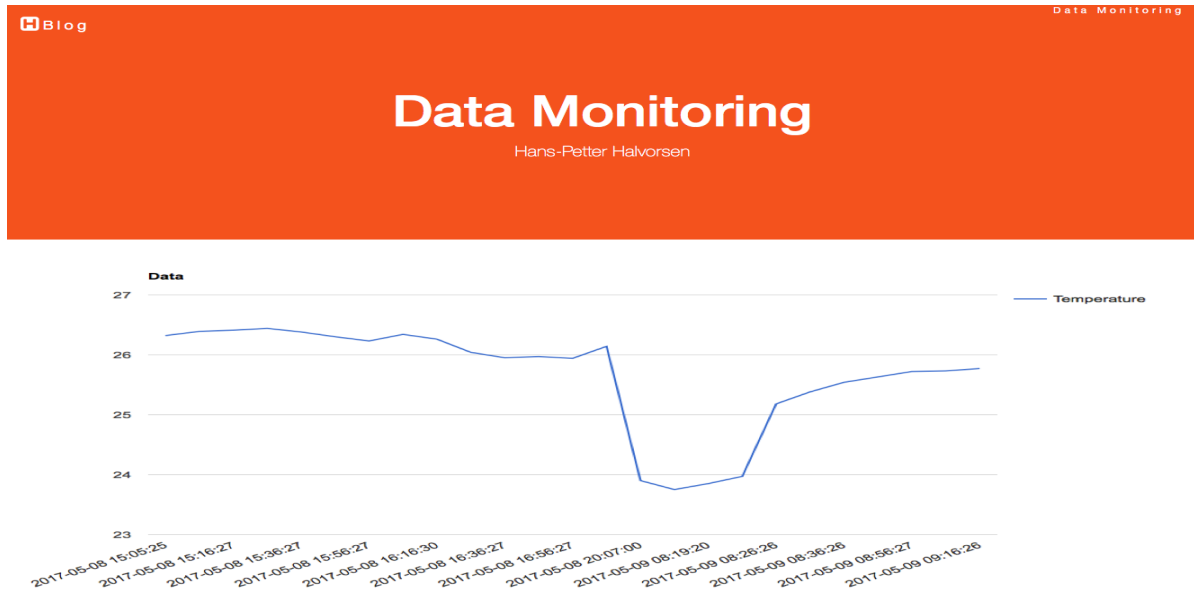


Figure 32-5: DMM Platform - Monitoring

With the DMM platform you can manage and monitor data in Control System, a SCADA System, a Home Automation solution, etc.

References

- Halvorsen, Hans-Petter (2017). Blog: *The Technical Guy*. Available: <https://www.halvorsen.blog>
- Halvorsen, Hans-Petter (2017). *Software Development - A Practical Approach!* ISBN: 978-82-691106-0-9. Available: <https://www.halvorsen.blog>
- Halvorsen, Hans-Petter (2017). *Structured Query Language*. Available: <https://www.halvorsen.blog>
- Nvidia (2017). *What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?* Available: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>
- Wikipedia (2017). *ASP.NET*. Available: <http://en.wikipedia.org/wiki/ASP.NET>
- Wikipedia (2017). *Deep Learning*. Available: https://en.wikipedia.org/wiki/Deep_learning
- Wikipedia (2017). *Industry 4.0*. Available: https://en.m.wikipedia.org/wiki/Industry_4.0
- Wikipedia (2017). *Scrum Development*. Available: [http://en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))



Hans-Petter Halvorsen

E-Mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>



<https://www.halvorsen.blog>

Industrial IT and Automation

A Practical Approach!

Hans-Petter Halvorsen

Copyright © 2017

ISBN: 978-82-691106-1-6

Publisher Identifier: 978-82-691106

<https://halvorsen.blog>