https://www.halvorsen.blog

Internet of Things and Arduino

Hans-Petter Halvorsen

Table of Contents

Introduction

- Cloud services and IoT solutions are becoming increasingly popular.
- Even the industry embrace IoT as Industrial Internet of Things (IIoT)
- IIoT is an important part of the next generation Automation Systems
- We will use Arduino as our IoT device
- Arduino is popular to use in different IoT applications

Topics

- Internet of Things (IoT)
- Microcontrollers (Arduino)
- PWM (Pulse Width Modulation)
- Automation
- ThingSpeak (IoT Cloud Service)
- Cyber Security

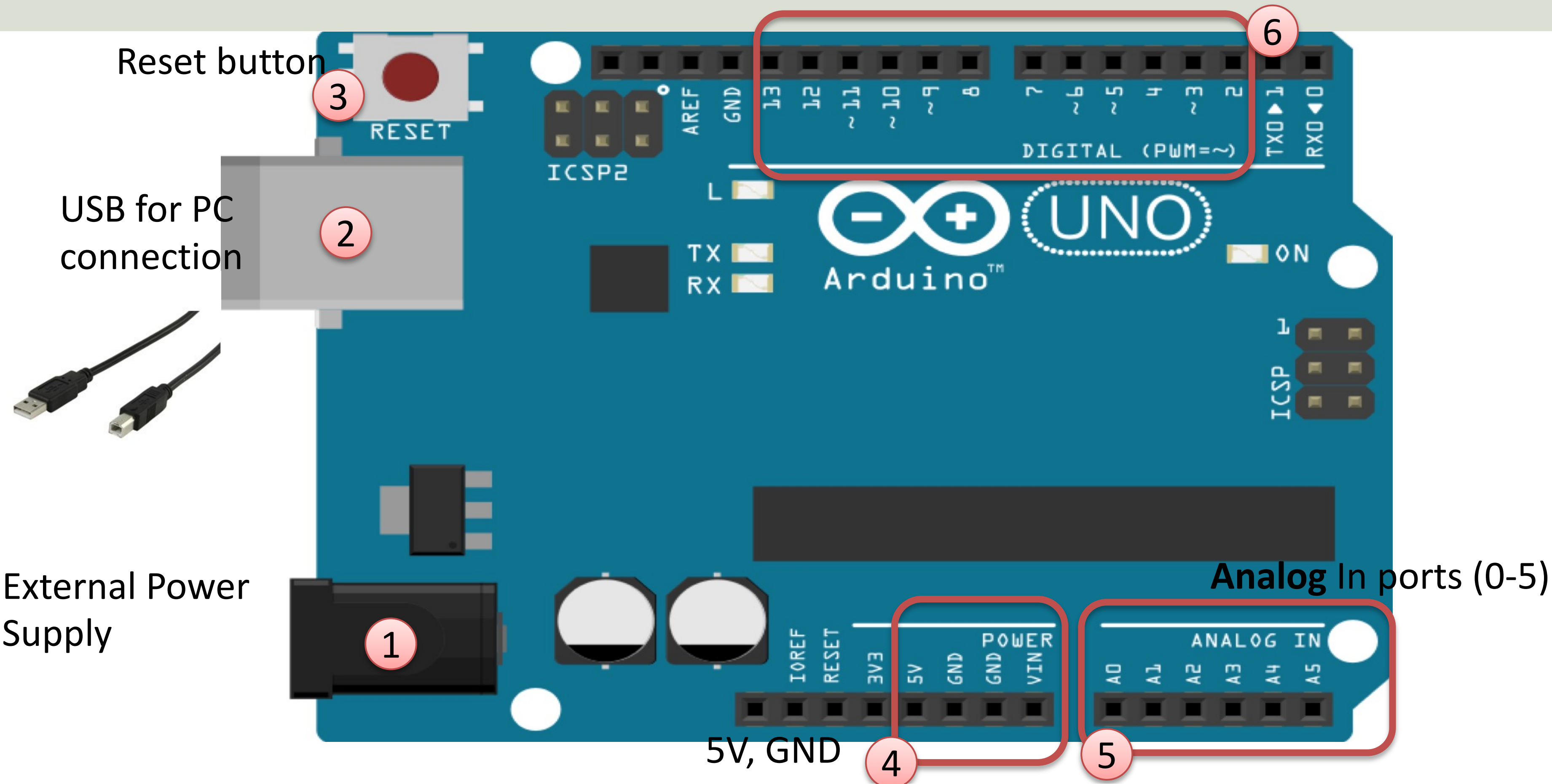
Delivery

- In this Assignment we will create an embedded Arduino PI(D) controller from scratch.
- One of the challenges is that Arduino UNO has no Analog Out.
- How can we solve that?
- The Data should be stored in the Cloud
- The Final System should be tested on the Air Heater System, i.e., you should control the Air Heater System
- Compare the results using LabVIEW LINX
- You should start your work by creating a System sketch. In that way you will get an overview of the system you are going to create and are able to plan your work and progress, so you are finished within the given deadline

For more details, see the web site

Arduino

Arduino


Digital ports (2-13)
6
Reset button
3
RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX0 1
RX0 0
DIGITAL (PWM=~)
ICSP2
L
USB for PC connection
2
UNO
TX
RX
Arduino™
ON
ICSP
1
External Power Supply
1
Analog In ports (0-5)
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
5V, GND
4
5

Arduino Software

Upload Code to Arduino Board

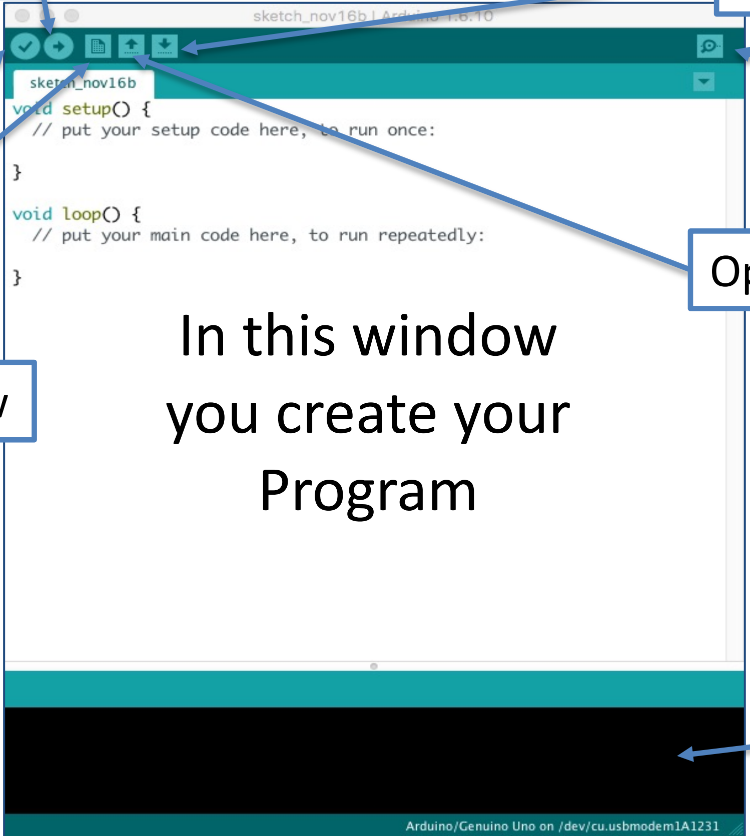
Save

Open Serial Monitor

Compile and Check if Code is OK

Open existing Code

Creates a New Code Window



In this window you create your Program

Error Messages can be seen here

An be downloaded for free:

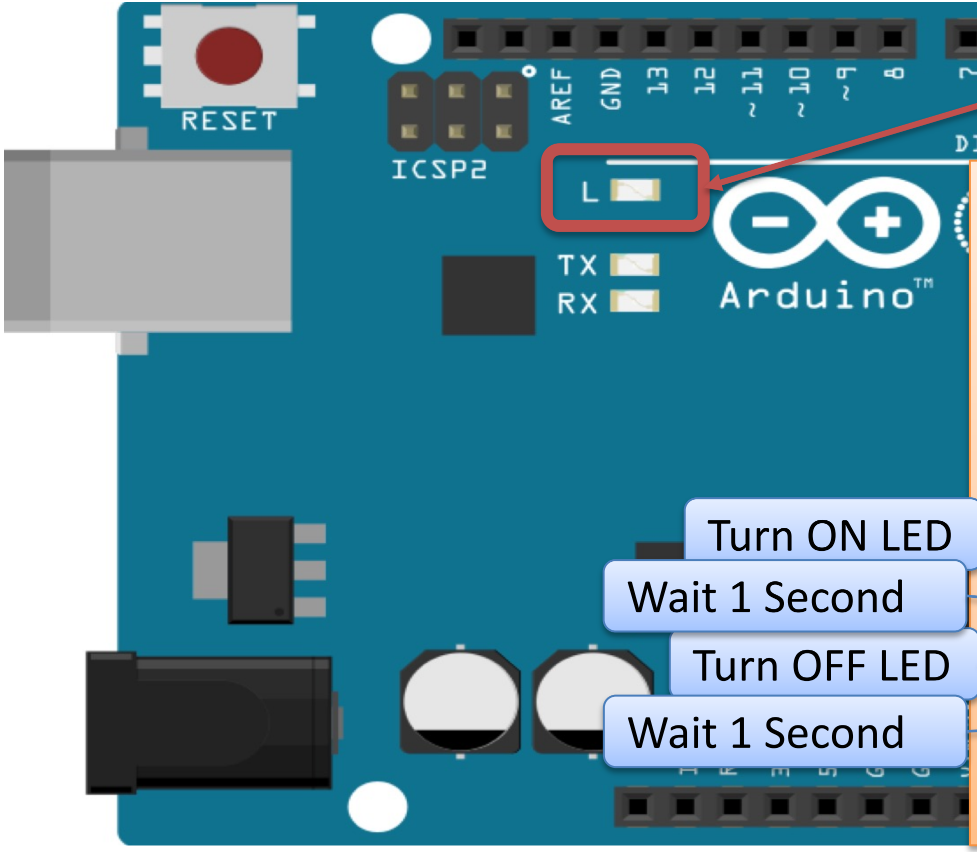
www.arduino.cc

Arduino Programs

All Arduino programs must follow the following main structure:

```
// Initialization, define variables, etc.

void setup()
{
        // Initialization
        ...
}

void loop()
{
        //Main Program
        ...
}
```

Blinking LED Example



Arduino UNO has a built-in LED that is connected to Port 13

```
void setup()
{
        pinMode(13, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

Turn ON LED

Wait 1 Second

Turn OFF LED

Wait 1 Second

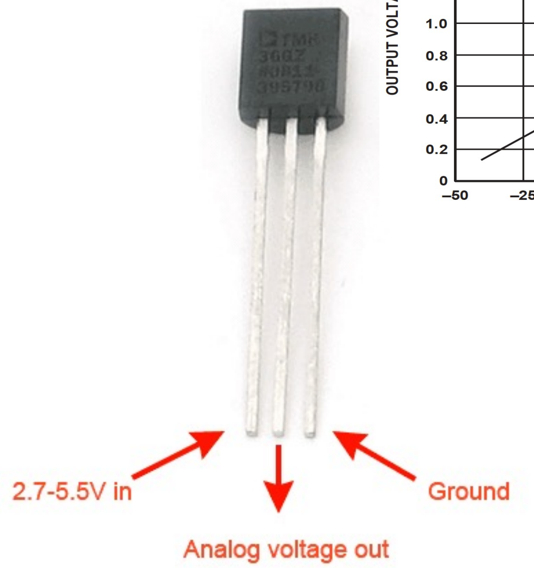
Temperature Sensors

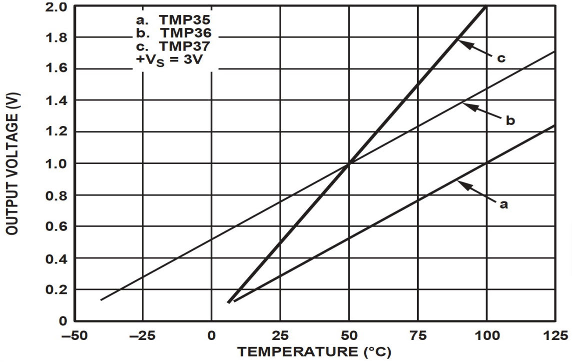
Hans-Petter Halvorsen

Table of Contents

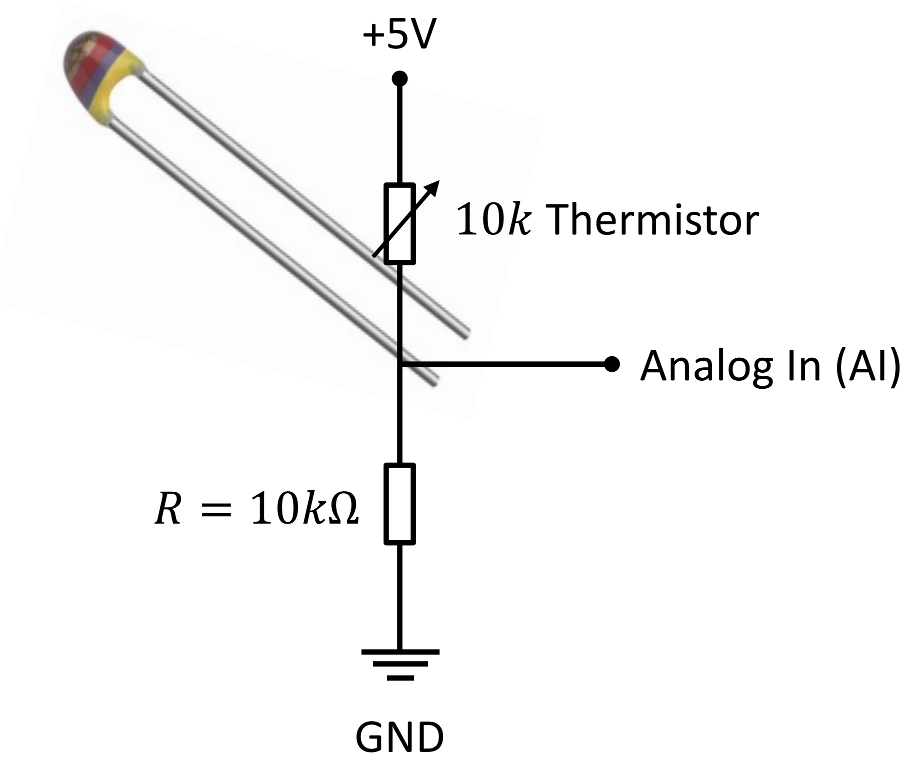
Temperature Sensors

TMP36 Temperature Sensor





10k Thermistor Temperature Sensor



Lowpass Filter

Hans-Petter Halvorsen

Lowpass Filter

The Transfer Function for a Low-pass filter is given by:

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

Where:
y is the Signal from the DAQ device (that contains noise)
y_f is the Filtered Signal
T_f is the Filter Time Constant

Why Lowpass Filter?

- In Measurement systems and Control Systems we typically need to deal with noise
- Noise is something we typically don't want
- Lowpass Filters are used to remove noise from the measured signals
- Noise is high-frequency signals
- A Lowpass Filter make sure the low frequencies pass (the measurements) and removes the high frequencies (the noise)

Discrete Lowpass Filter

Lowpass Filter:

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

We can find the Differential Equation for this filter using Inverse Laplace:

$$T_f \dot{y}_f + y_f = y$$

We use Euler Backward method: $\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}$

Then we get:

$$T_f \frac{y_f(k) - y_f(k-1)}{T_s} + y_f(k) = y(k)$$

This gives: $y_f(k) = \frac{T_f}{T_f + T_s} y_f(k-1) + \frac{T_s}{T_f + T_s} y(k)$

We define:

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

Finally, we get the following discrete version of the Lowpass Filter:

$$y_f(k) = (1 - a) y_f(k-1) + a y(k)$$

This equation can easily be implemented using the Arduino software or another programming language

PID Controller

Hans-Petter Halvorsen

Table of Contents

PID Controller

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

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

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

r is the Reference Signal or Set-point

y is the Process value, i.e., the Measured value

Tuning Parameters:

K_p Proportional Gain

T_i Integral Time [sec.]

T_d Derivative Time [sec.]

Discrete PI controller

We start with the continuous PI Controller:

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

We derive both sides in order to remove the Integral:

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

We can use the Euler Backward Discretization method:

$$\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}$$

Where T_s is the Sampling Time

Then we get:

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

Finally, we get:

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

Where $e_k = r_k - y_k$

Alternative PI controller

We can also put the PI Controller on Transfer Function form (we use Laplace):

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

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

This gives:

$$\dot{z} = e$$

$$u = K_p e + \frac{K_p}{T_i} z$$

This is the PI controller on State-space form

Using Euler, we get the following discrete PI controller:

$$e_k = r_k - y_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

$$z_{k+1} = z_k + T_s e_k$$

This algorithm can easily be implemented in the Arduino software.

Arduino Analog Out

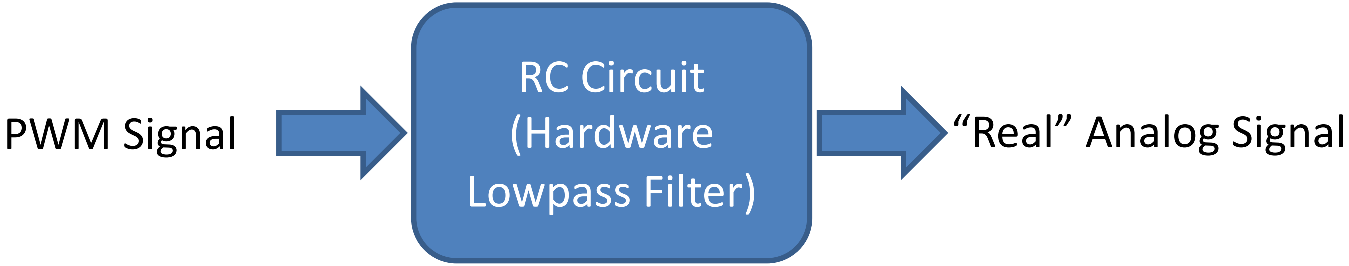
The Output (typically 0-5V) of the PI(D) controller should be sent to the process.

Arduino UNO has no Analog Output Pins

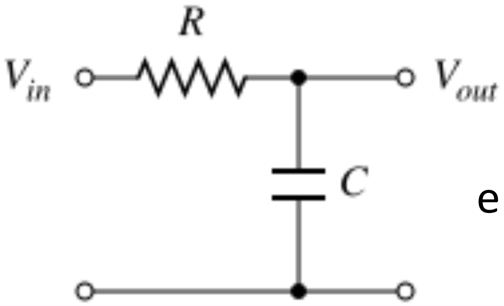
Solutions:

- Smooth PWM output using RC Circuit
- DAC chip (Digital to Analog Converter)

Smooth PWM output using RC Circuit



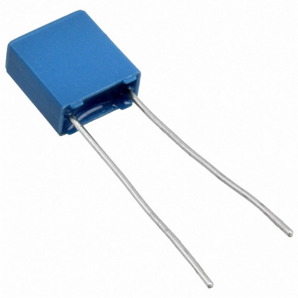
e.g., $R = 3.9k\Omega$



e.g., $C = 10\mu F$

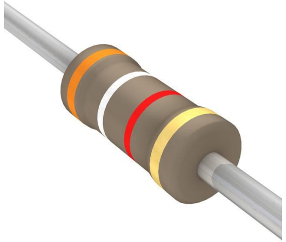
Electrical Components

Capacitor



e.g., $C = 10\mu F$

Resistor



$R = 3.9k\Omega$

A capacitor stores and releases electrical energy in a circuit. When the circuits voltage is higher than what is stored in the capacitor, it allows current to flow in, giving the capacitor a charge. When the circuits voltage is lower, the stored charge is released. Often used to smooth fluctuations in voltage

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

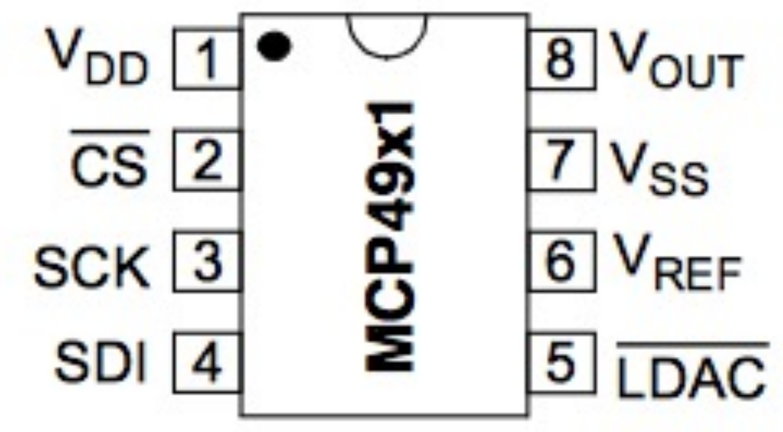
A resistor resists the flow of electrical energy in a circuit, changing the voltage and current as a result (according to Ohms law, $U = RI$). Resistor values are measured in ohms (Ω). The color stripes on the sides of the resistor indicate their values. You can also use a Multi-meter in order to find the value of a given resistor.

These electronics components are typically included in a “Starter Kit”, or they can be bought “everywhere” for a few bucks.

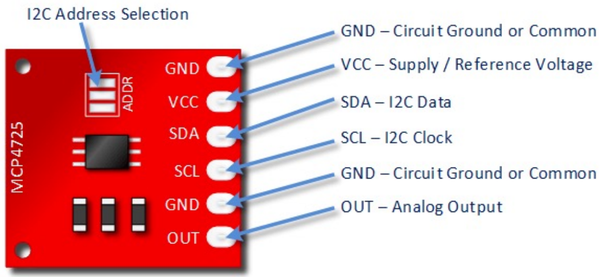
DAC Chip

Arduino UNO has no Analog Output Pins, so we need a DAC such as, e.g., Microchip **MCP4911**, MCP4725 or similar

MCP4911: 10-bit single DAC, SPI Interface



MCP4725



12-bit resolution
I2C Interface

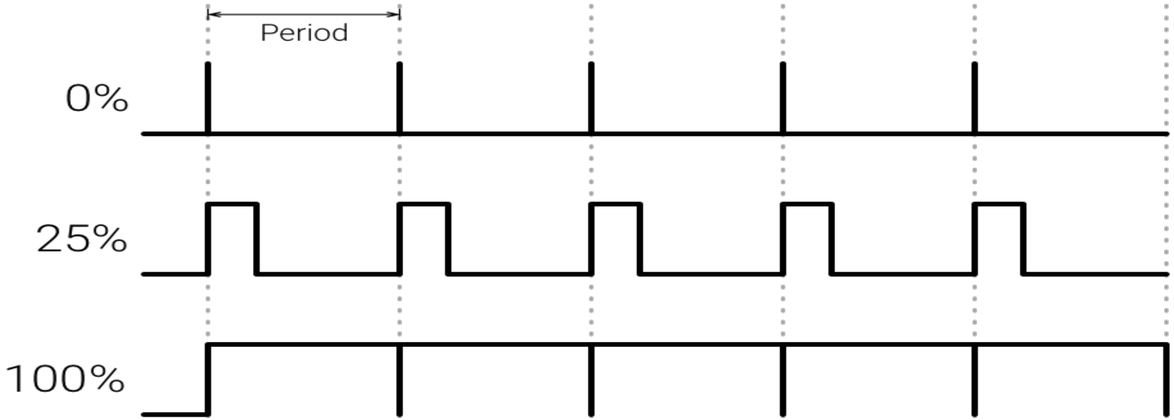
The MCP4725 is a little more expensive, but simpler to use

Microchip MCP4911 can be bought "everywhere" (10 NOK).

PWM

PWM is a digital (i.e., square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



Arduino Library

Hans-Petter Halvorsen

Table of Contents

Arduino Library

Why create your own Libraries?

- Better Code structure
- Reuse your Code in different Applications
- Distribute to others

You need at least two files for a library:

- Header file (.h) - The header file has definitions for the library
- Source file (.cpp) – The Functions within the Class

Note the Library Name, Folder name, .h and .cpp files all need to have the same name

Arduino Library Example

Fahrenheit.h

```cpp
/*
  Fahrenheit.h - Library converting between Celsius and Fahrenheit.
  Created by Hans-Petter Halvorsen, 2018
*/
#ifndef Fahrenheit_h
#define Fahrenheit_h

#include "Arduino.h"

class Fahrenheit{
  public:
    Fahrenheit();
    float c2f(float Tc);
    float f2c(float Tf);
};

#endif
```

Fahrenheit.cpp

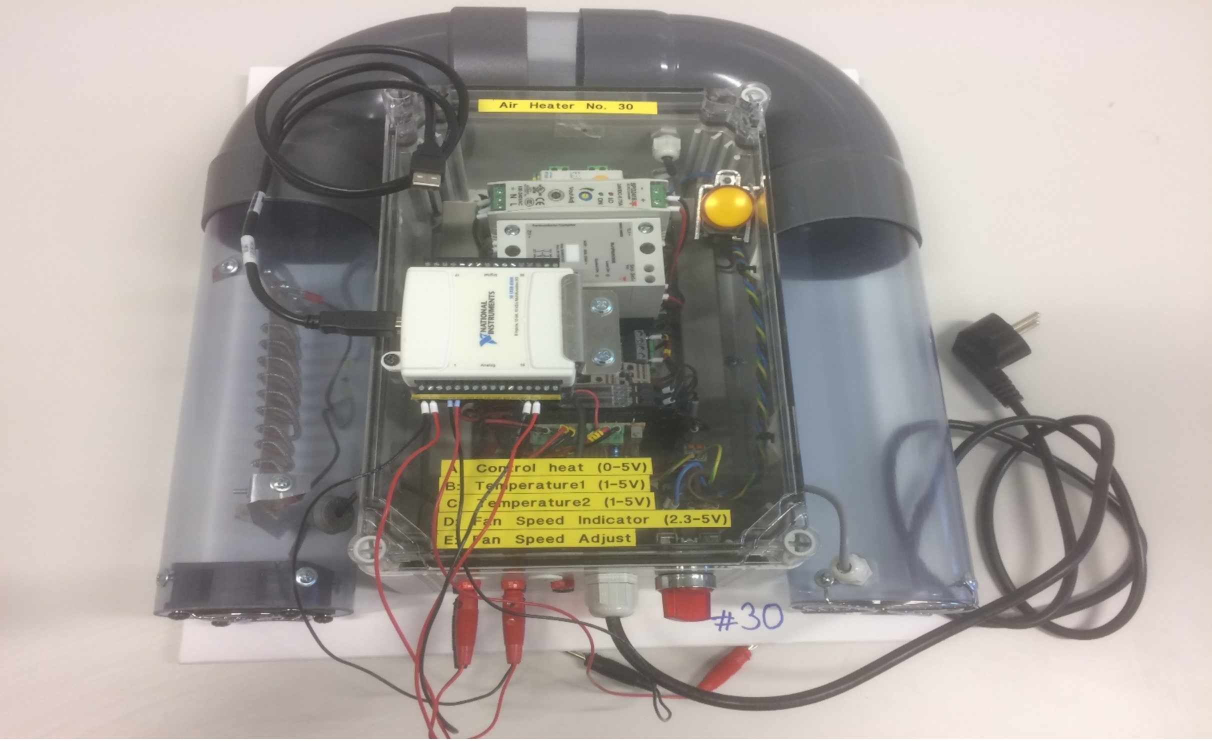
```cpp
/*
  Fahrenheit.cpp - Library converting between Celsius ar
  Created by Hans-Petter Halvorsen, 2018
*/

#include "Fahrenheit.h"

Fahrenheit::Fahrenheit(){

}

float Fahrenheit::c2f(float Tc){
  float Tf;
  Tf = Tc * 9/5 + 32;
  return Tf;
}

float Fahrenheit::f2c(float Tf){
  float Tc;
  Tc = (Tf-32)*(5/9);
  return Tc;
}
```

```cpp
#include <Fahrenheit.h>

Fahrenheit fahr;

void setup()
{
  float f;
  float c;

  Serial.begin(9600);
}

void loop()
{
  ...
  f = fahr.c2f(c);
  Serial.println(f);

  ...
  c = fahr.f2c(f);
  Serial.println(c);
}
```

Air Heater

Hans-Petter Halvorsen

Air Heater System



We can, e.g., use the following values in the simulation:

$$\theta_t = 22\ s$$

$$\theta_d = 2\ s$$

$$K_h = 3.5\ \frac{°\mathrm{C}}{V}$$

$$T_{env} = 21.5\ °\mathrm{C}$$

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

Discrete Air Heater

Continuous Model:

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

We can use e.g., the Euler Approximation in order to find the discrete Model:

$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}$$

T_s - Sampling Time

$x(k)$ - Present value

$x(k+1)$ - Next (future) value

The discrete Model will then be on the form:

$$x(k+1) = x(k) + \ ...$$

We can then implement the discrete model in any programming language

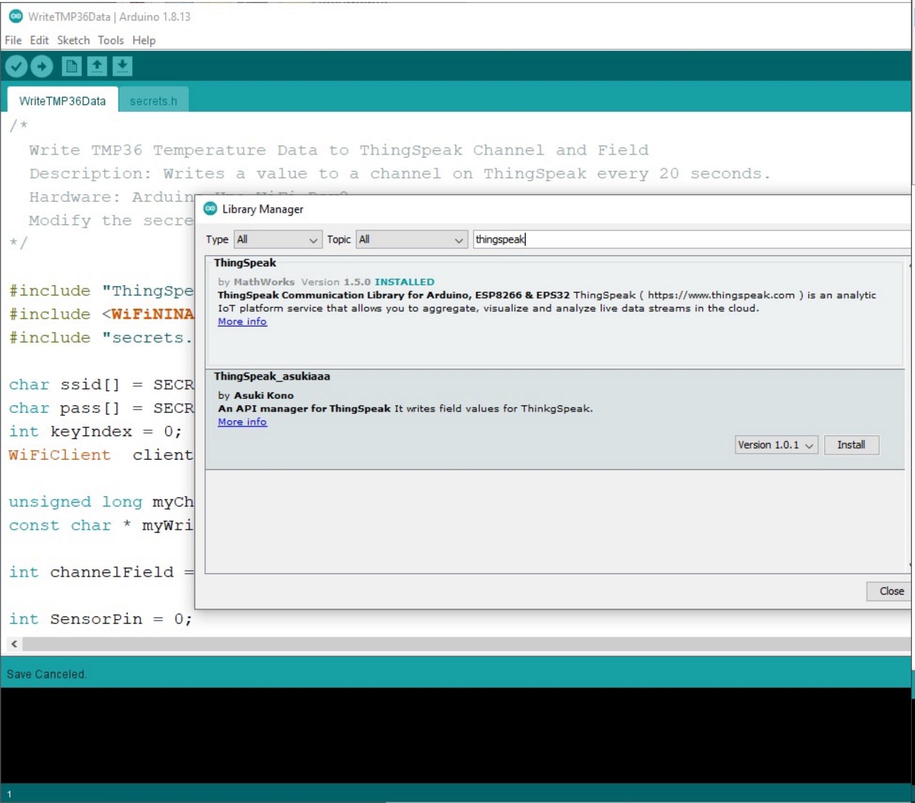
ThingSpeak

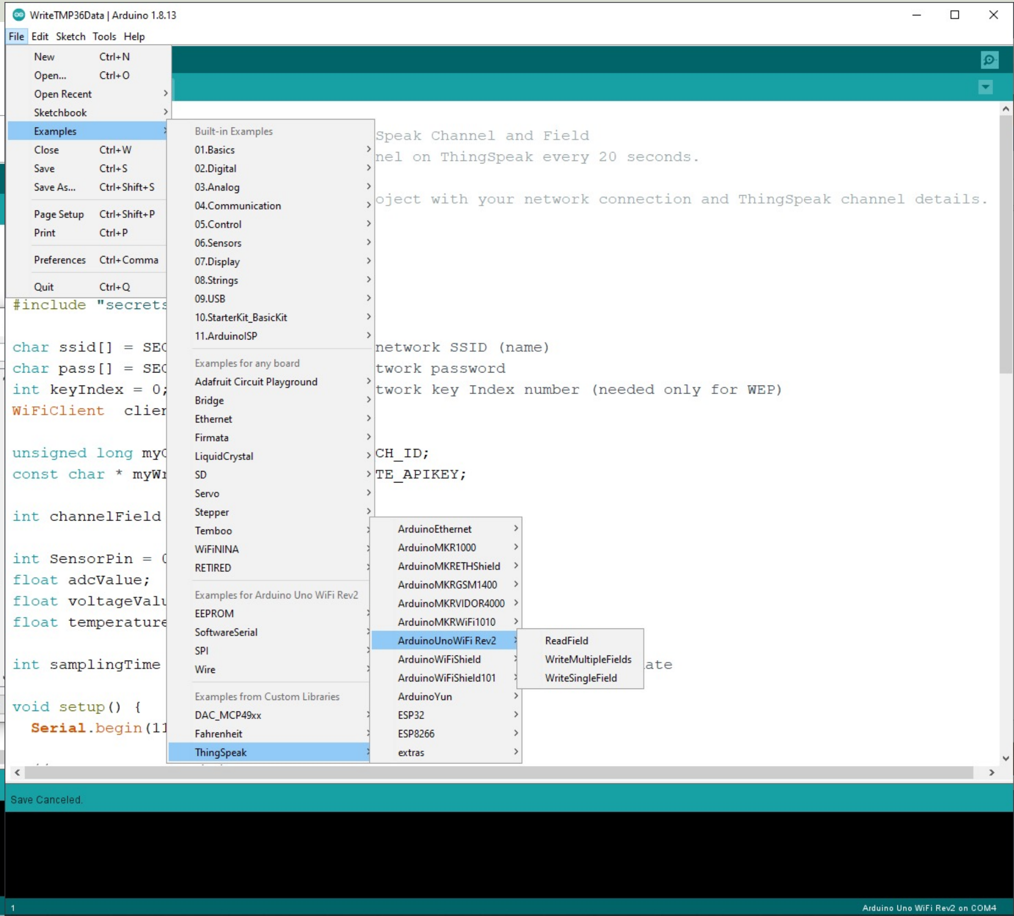
ThingSpeak

- ThingSpeak is an IoT analytics platform service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- ThingSpeak has a free Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- It works with Arduino, Raspberry Pi, MATLAB and LabVIEW, Python, etc.

https://thingspeak.com

ThingSpeak + Arduino





ThingSpeak + Arduino

- Install the “thingspeak“ Arduino Library using the Library Manager in your Arduino IDE
- Use e.g., the built-in example "WriteSingleField" as a starting point.
- This example is available for different boards and configuration, such as Arduino WiFi rev2 board, Arduino WiFi shield, etc.
- Then you can modify the example to suit your needs

Currently, a single channel can only be **updated once every 15 seconds**.

ThingSpeak + Arduino

This Example uses an Arduino WiFi rev2 board.

The Example reads values from TMP36 Temperature Sensor and write the values to ThingSpeak

```
#include "ThingSpeak.h"
#include <WiFiNINA.h>
#include "secrets.h"
char ssid[] = SECRET_SSID;    //  your network SSID (name)
char pass[] = SECRET_PASS;   // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;
unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
int channelField = 3;
int SensorPin = 0;
float adcValue;
float voltageValue;
float temperatureValue = 0;
int samplingTime = 20000; // Wait 20 seconds between each hannel update
void setup() {
  Serial.begin(115200);  // Initialize serial
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }
String fv = WiFi.firmwareVersion();
  if (fv != "1.0.0") {
    Serial.println("Please upgrade the firmware");
  }

  ThingSpeak.begin(client);  //Initialize ThingSpeak
}
void loop() {
 // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
  adcValue = analogRead(SensorPin); // Get Data from Temperature Sensor
  voltageValue = (adcValue*5)/1023;
  temperatureValue = 100*voltageValue - 50;
  Serial.println(temperatureValue);

  // Write to ThingSpeak
  int x = ThingSpeak.writeField(myChannelNumber, channelField, temperatureValue, myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }
 delay(20000); // Wait 20 seconds to update the channel again
}
```

secrets.h

```
// Use this file to store all of the private credentials
// and connection details

#define SECRET_SSID "MySSID"      // replace MySSID with your WiFi network name
#define SECRET_PASS   "xxxxxx"    // replace MyPassword with your WiFi password

#define SECRET_CH_ID 000000       // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY   "XYZ"      // replace XYZ with your channel write API Key
```

Read/Write using a Web Browser

Field 3 Kp ☑

Set Kp Remotely Example:

Enter the following in a Web Browser (or from a Programming Language)

We set Kp=2

https://api.thingspeak.com/update?api_key=<WriteKey>&field3=2

Read Kp Remotely Example:

https://api.thingspeak.com/channels/<ChannelId>/fields/3/last.json?key=<ReadKey>

Response in Browser: {"created_at":"2017-06-26T07:41:54Z","entry_id":1270,"field3":"2"}

We read Kp=2

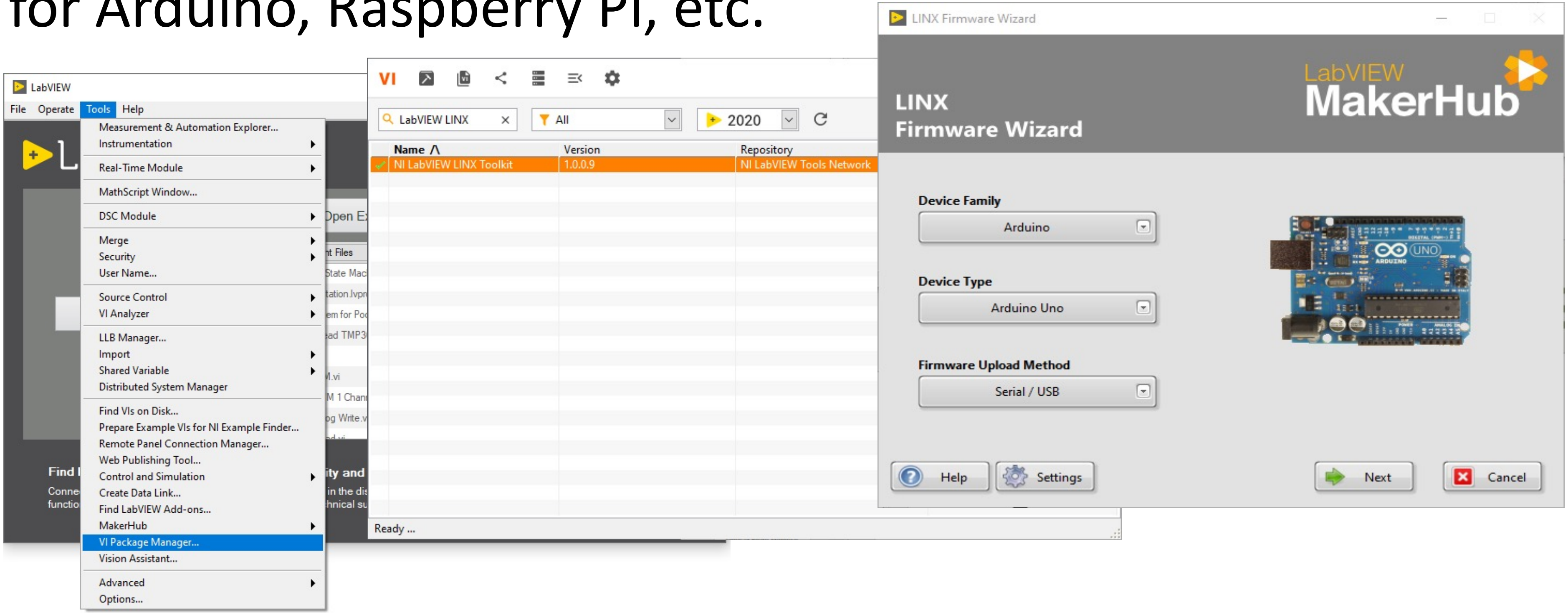
LabVIEW LINX

Hans-Petter Halvorsen

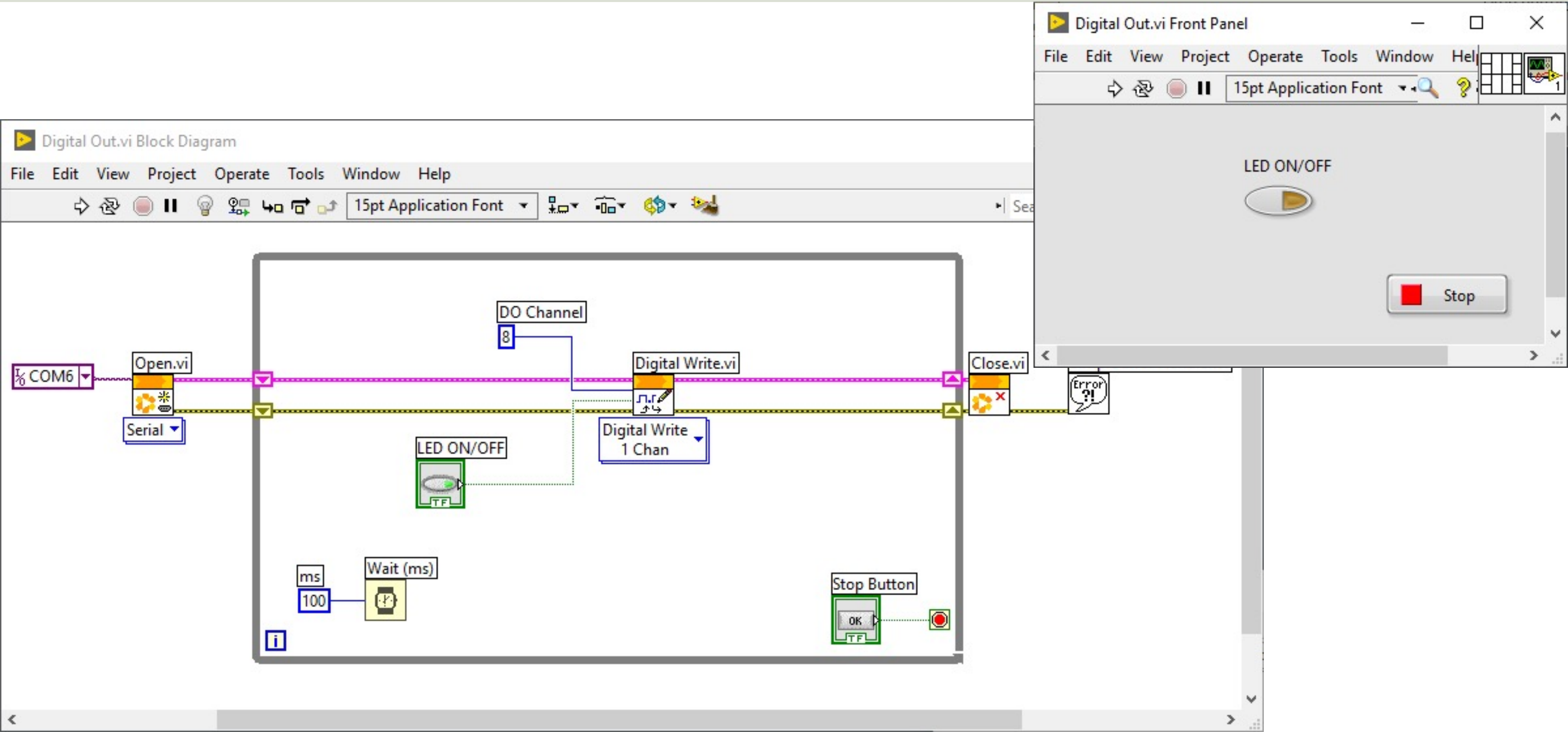
Table of Contents

LabVIEW LINX

The LabVIEW LINX Toolkit adds support for Arduino, Raspberry Pi, etc.

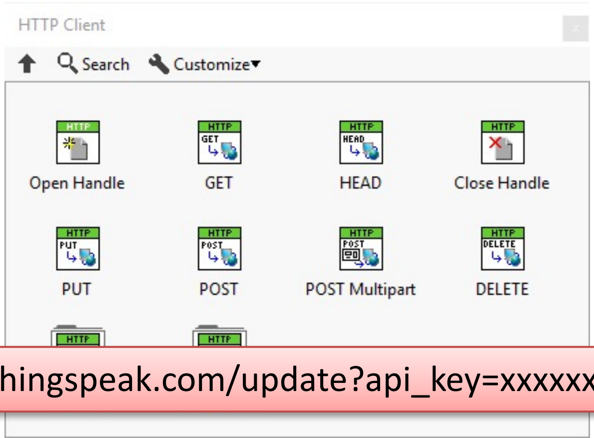


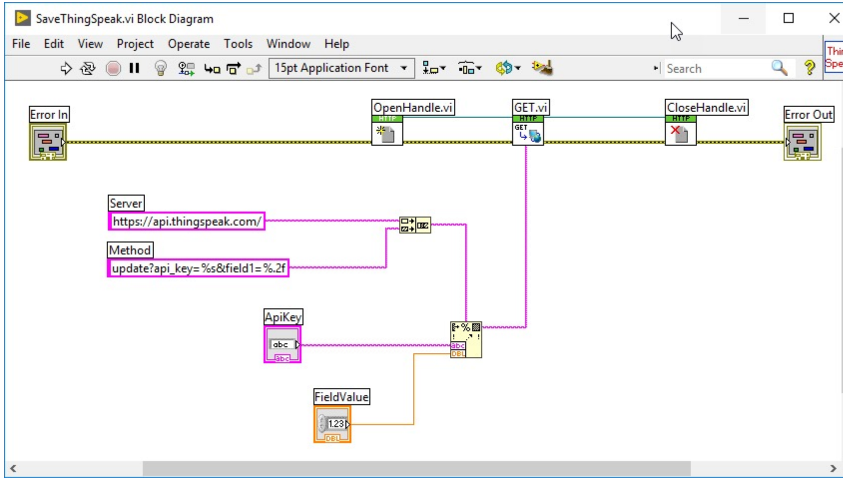
LabVIEW LINX Example



ThingSpeak + LabVIEW

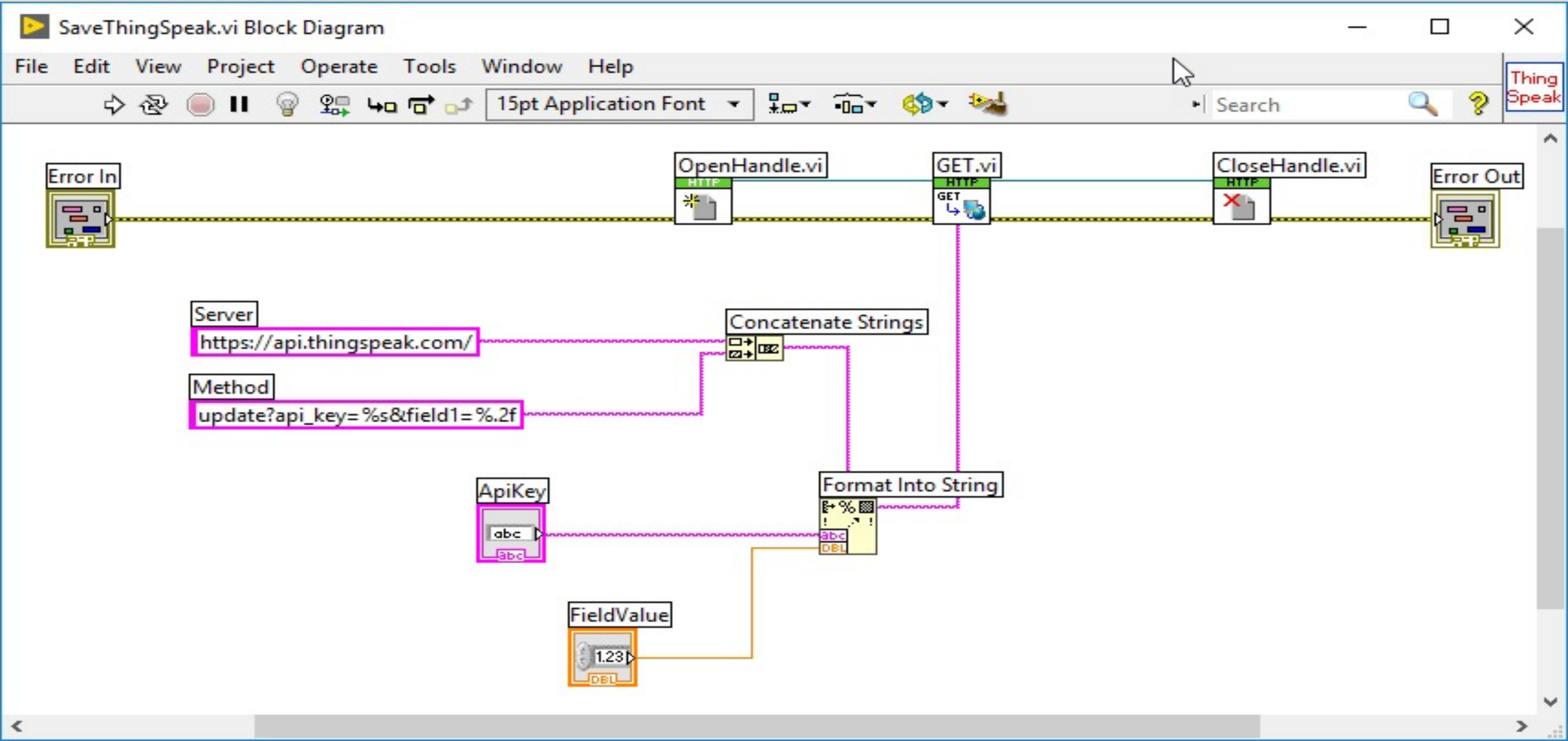
- ThingSpeak uses standard HTTP REST API, which can be used from any kind of Programming Language, including LabVIEW
- In LabVIEW you can use the HTTP client VIs





https://api.thingspeak.com/update?api_key=xxxxxxx&field1=22.5

ThingSpeak + LabVIEW



Cyber Security

Hans-Petter Halvorsen

Table of Contents

Cyber Security and IoT

- IoT solutions and Data Security? How can we make sure our applications and data are safe?
- Security is crucial in IoT/IIoT Applications. Why?
- What issues do we need to deal with regarding IoT and Cyber Security?
- What can be (or what have you) done to protect the system (and data) you have created?
- How does ThingSpeak handle security?
- Etc.

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog