

<https://www.halvorsen.blog>



# Arduino and ThingSpeak

Hans-Petter Halvorsen

# Table of Contents

- Arduino
- ThingSpeak
- Arduino + ThingSpeak
- Arduino Example
  - Write Data to ThingSpeak using a TMP36 Temperature Sensor



# Arduino

Hans-Petter Halvorsen

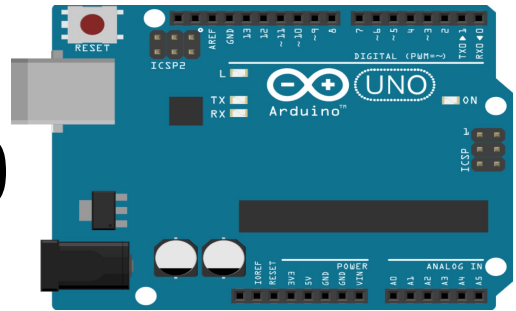
[Table of Contents](#)

# Arduino

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- It's intended for anyone making interactive projects, from kids to grown-ups.
- You can connect different Sensors, like Temperature, etc.
- It is used a lots in Internet of Things projects
- Homepage:  
<https://www.arduino.cc>

# Arduino

- Arduino is a Microcontroller
- Arduino is an open-source platform with Input/Output Pins (Digital In/Out, Analog In and PWM)
- Price about \$20
- Arduino Starter Kit ~\$40-80  
with Cables, Wires, Resistors, Sensors, etc.

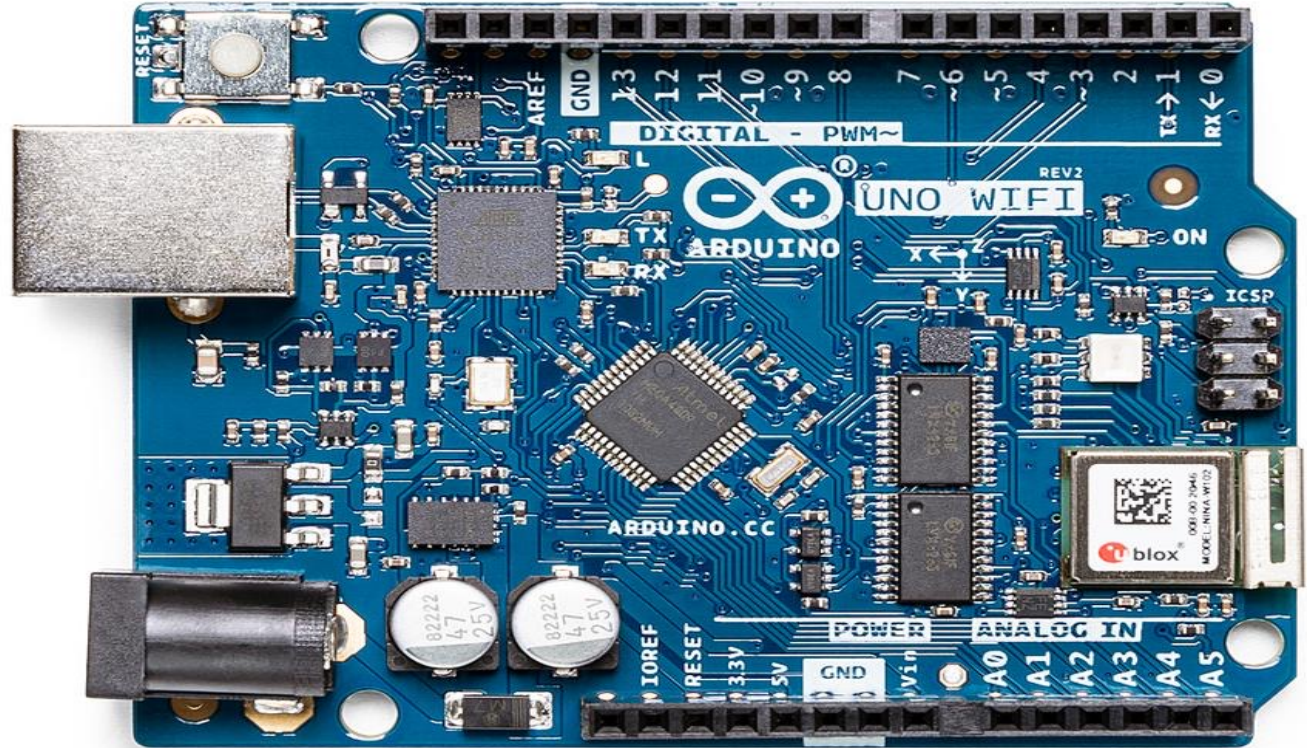


# Arduino UNO WiFi Rev 2

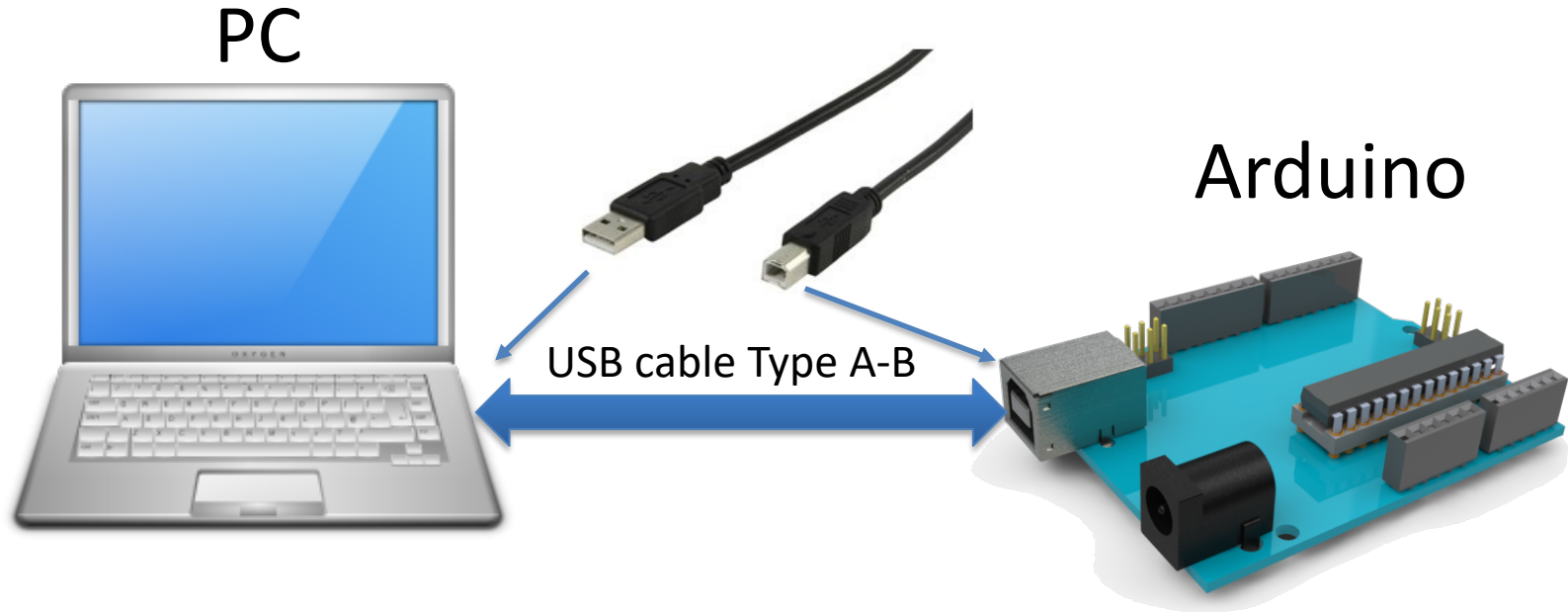
- Lots of different Arduino boards exists
- The basic Arduino UNO don't have WiFi or Ethernet
- We need to use a board with built-in WiFi or Ethernet
- Or we can use a WiFi or Ethernet Shield
- In this Tutorial an “Arduino UNO WiFi Rev 2” is used

# Arduino UNO WiFi Rev 2

The Arduino Uno WiFi is functionally the same as the Arduino Uno Rev3, but with the addition of WiFi / Bluetooth and some other enhancements.



# Connect Arduino to your PC





# Arduino Software

Upload Code to Arduino Board

Save

Open Serial Monitor

Compile and Check  
if Code is OK

Open existing Code

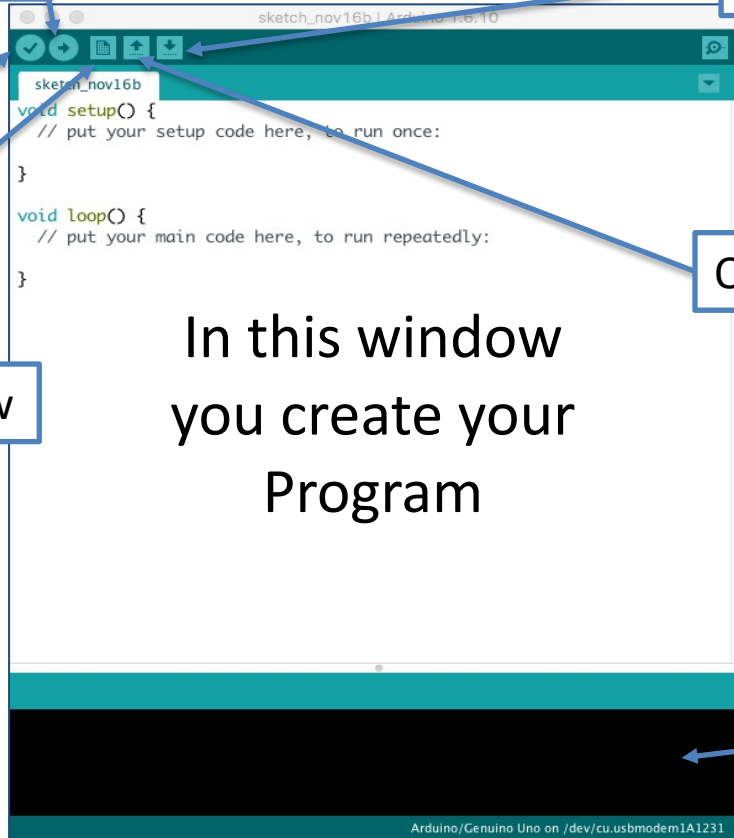
Creates a New Code Window

In this window  
you create your  
Program

The software can be  
downloaded for free:

[www.arduino.cc](http://www.arduino.cc)

Error Messages  
can be seen here



# Arduino Programs

All Arduino programs must follow the following main structure:

```
// Initialization, define variables, etc.  
  
void setup()  
{  
    // Initialization  
    ...  
}  
  
void loop()  
{  
    //Main Program  
    ...  
}
```



# 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 (IoT) 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

## Work

Channel ID:            |  temperature  
Author:             
Access: Public

Private View **Public View** Channel Settings Sharing API Keys Data Import / Export

[+ Add Visualizations](#) [+ Add Widgets](#) [Export recent data](#)

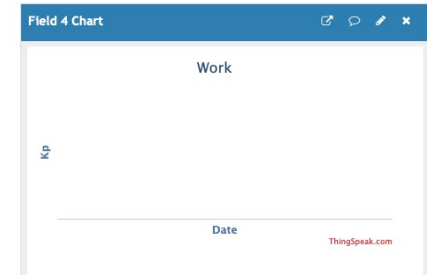
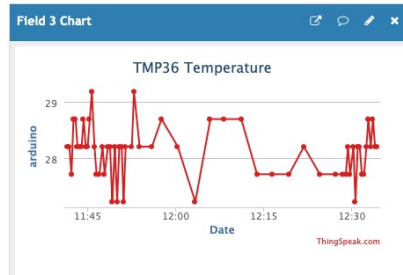
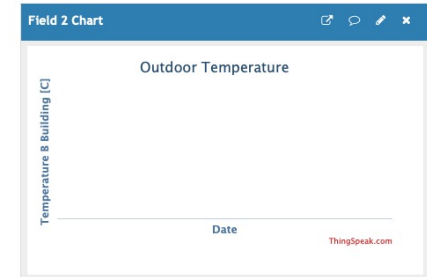
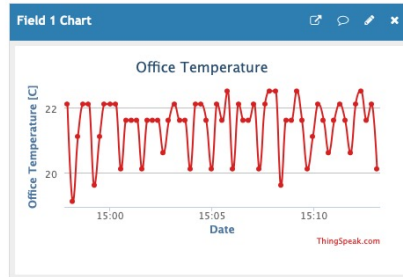
[MATLAB Analysis](#) [MATLAB Visualization](#)

[More Information](#)

Channel 1 of 3 < >

## Channel Stats

Created: 4 years ago  
Last entry: less than a minute ago  
Entries: 242



<https://thingspeak.com>



# Arduino + ThingSpeak

Hans-Petter Halvorsen

[Table of Contents](#)

# 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

The image shows the Arduino IDE interface with the following components:

- Sketch Editor:** Contains the code for writing temperature data to a ThingSpeak channel. The code includes comments describing the purpose and hardware, and defines variables for network credentials, channel ID, and sensor pin.
- Library Manager:** Opened to search for and install the 'ThingSpeak' library by Asuki Kono. The library is shown as installed.
- File Menu:** Opened to show the 'Examples' submenu, which is further expanded to show the 'ThingSpeak' example category.
- Code Editor:** Shows the implementation of the sketch, including the inclusion of the ThingSpeak library and the definition of variables for network SSID, password, channel ID, and sensor pin.

```
/*
 * Write TMP36 Temperature Data to ThingSpeak Channel and Field
 * Description: Writes a value to a channel on ThingSpeak every 20 seconds.
 * Hardware: Arduino Uno
 * Modify the secrets.h file to match your hardware.
 */

#include "ThingSpeak.h"
#include <WiFiNINA.h>
#include "secrets.h"

char ssid[] = SECRET_NETWORK_SSID;
char pass[] = SECRET_NETWORK_PASSWORD;
int keyIndex = 0;
WiFiClient client;

unsigned long myChannelID;
const char * myWriteField;

int channelField = 0;

int SensorPin = 0;

void setup() {
  Serial.begin(115200);
}
```





# Arduino Example

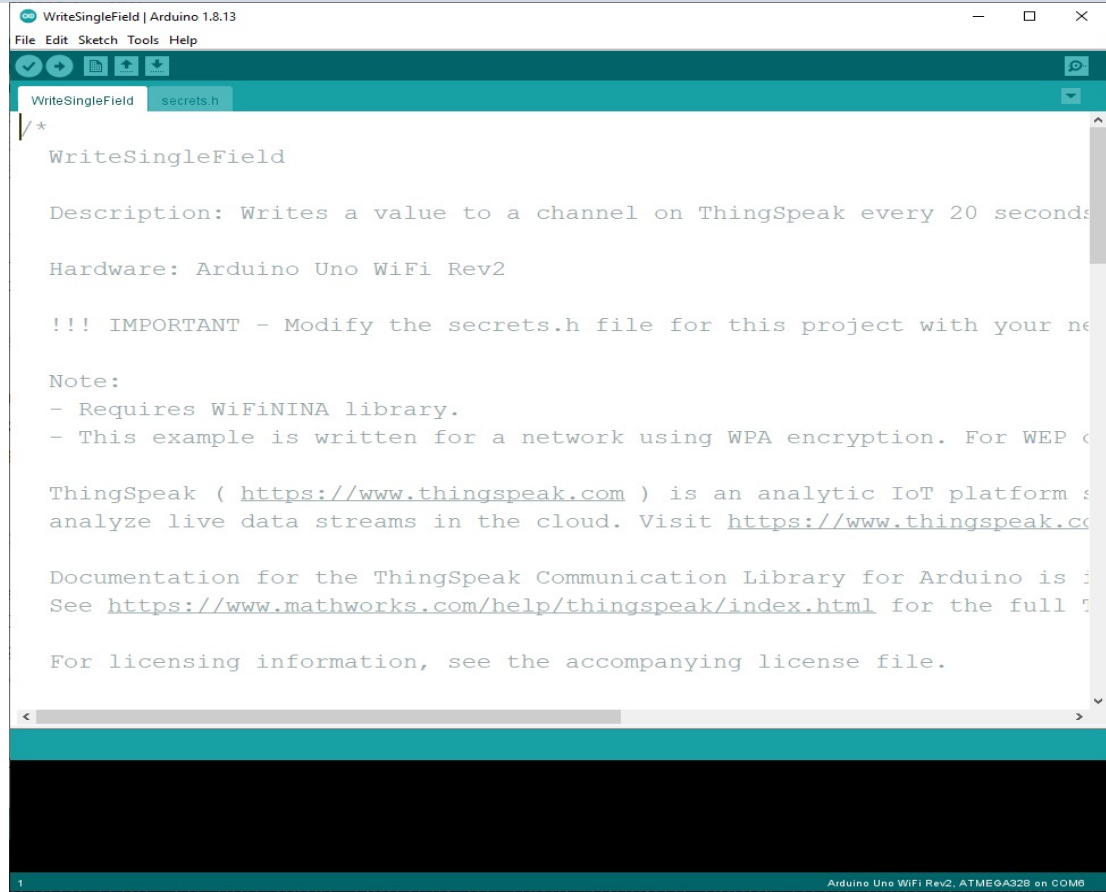
# WriteSingleField Example

We use the the “WriteSingleField” Example as a starting point.

We just need to change WiFi information, like Password, etc.

Then we change ThingSpeak Information.

Finally, we add code for reading Temperature values from the TMP36 Temperature Sensor



The screenshot shows the Arduino IDE interface for the 'WriteSingleField' example. The title bar indicates 'WriteSingleField | Arduino 1.8.13'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for saving, undo, redo, and other standard IDE functions. The main editor area displays the following text:

```
WriteSingleField

Description: Writes a value to a channel on ThingSpeak every 20 seconds

Hardware: Arduino Uno WiFi Rev2

!!! IMPORTANT - Modify the secrets.h file for this project with your ne

Note:
- Requires WiFiNINA library.
- This example is written for a network using WPA encryption. For WEP c

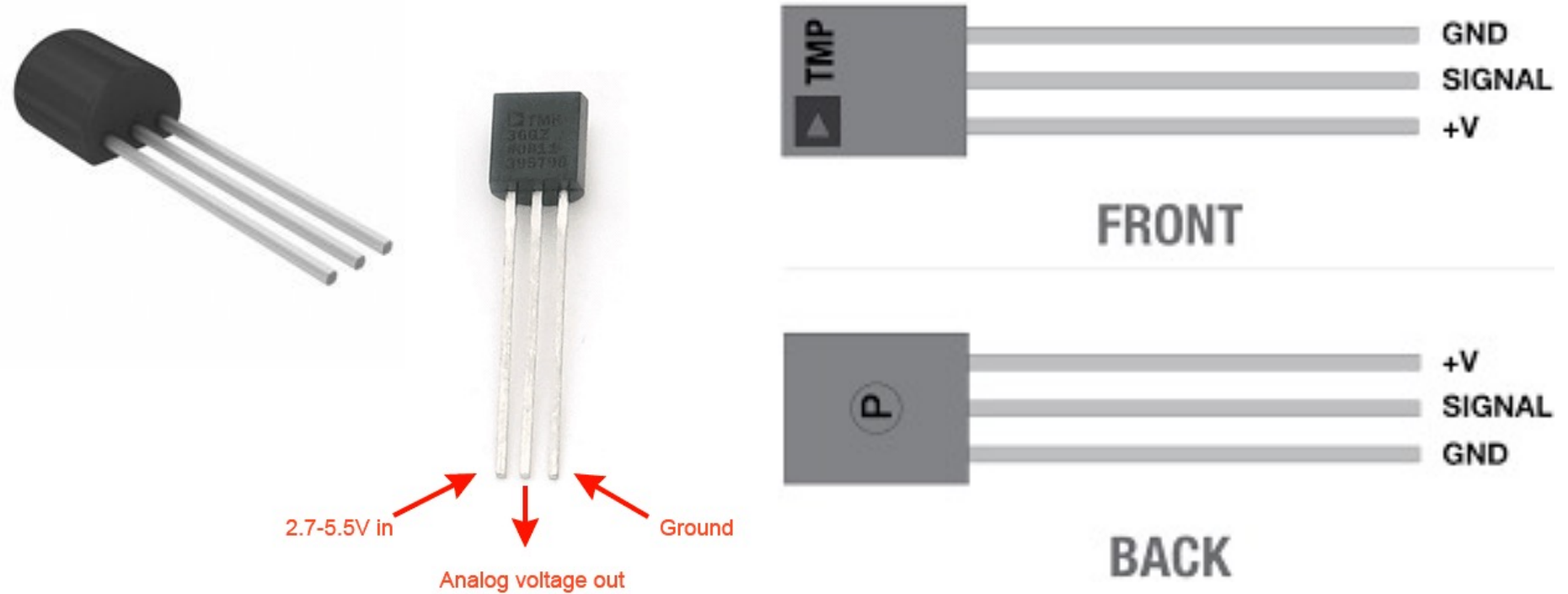
ThingSpeak ( https://www.thingspeak.com ) is an analytic IoT platform s
analyze live data streams in the cloud. Visit https://www.thingspeak.co

Documentation for the ThingSpeak Communication Library for Arduino is :
See https://www.mathworks.com/help/thingspeak/index.html for the full

For licensing information, see the accompanying license file.
```

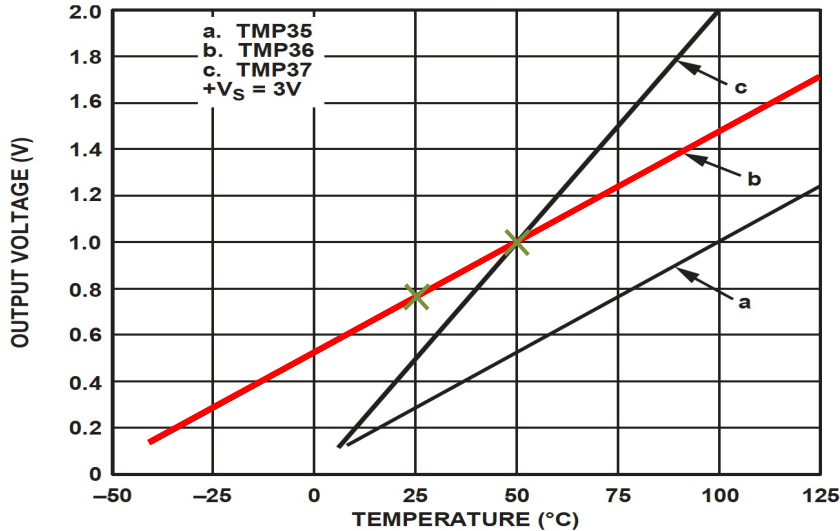
At the bottom of the IDE, a status bar shows '1' on the left and 'Arduino Uno WiFi Rev2, ATMEGA328 on COM9' on the right.

# TMP36



TMP is a small, low-cost temperature sensor and cost about \$1 (you can buy it “everywhere”)

# Linear Scaling



This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75} (x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

Convert from Voltage (V) to degrees Celsius  
From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25^{\circ}C)$$
$$(x_2, y_2) = (1V, 50^{\circ}C)$$

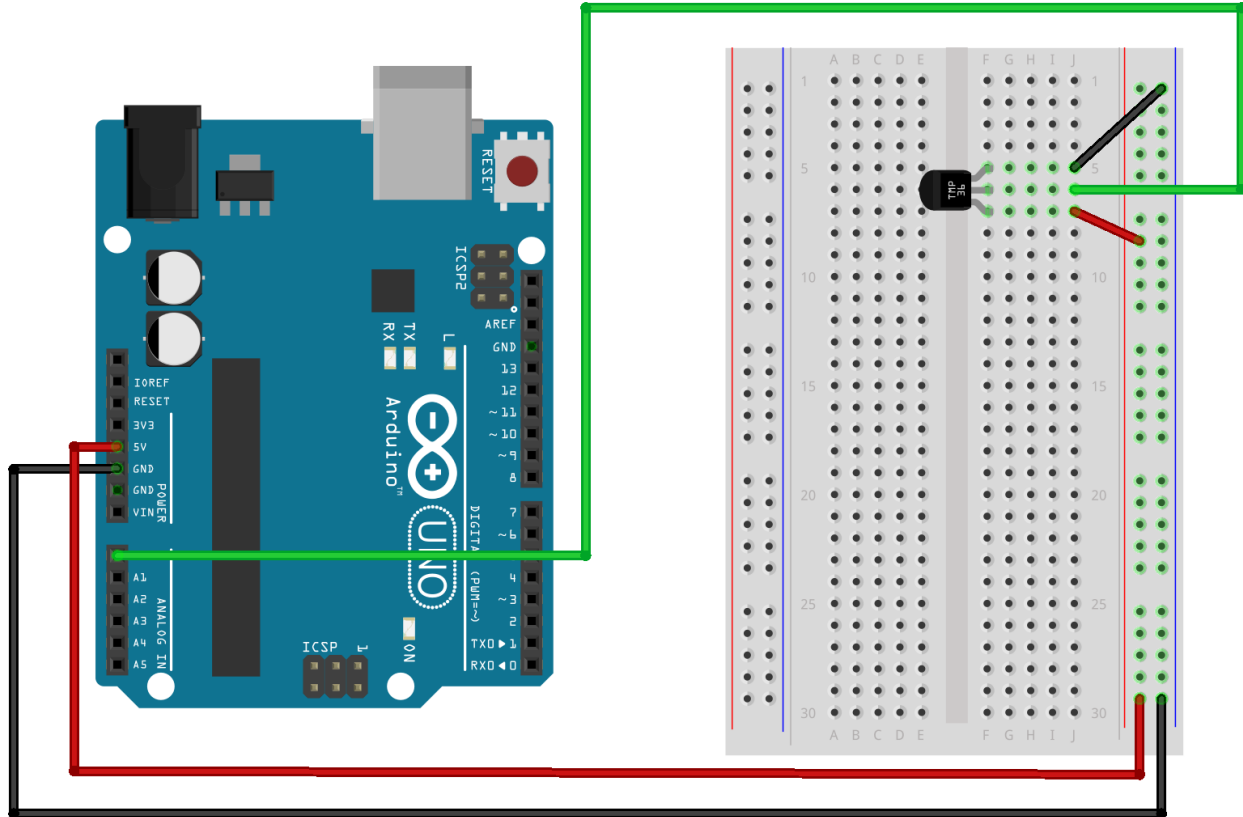
There is a linear relationship between  
Voltage and degrees Celsius:

$$y = ax + b$$

We can find a and b using the following  
known formula:

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

# Wiring



# Temperature Conversion

We want to present the value from the sensor in degrees Celsius:

1. The function `analogRead()` gives a value between 0 and 1023 (Arduino UNO has a built-in 10-bit ADC,  $2^{10}=1024$ )
2. Then we convert this value to 0-5V.
3. Finally, we convert to degrees Celsius using information from the Datasheet presented on the previous page ( $y = 100x - 50$ )
4. The we can, e.g., show the Temperature value in the Serial Monitor

# Code

```
WriteTMP36Data | Arduino 1.8.13
File Edit Sketch Tools Help
WriteTMP36Data secrets.h
/*
  Write TMP36 Temperature Data to ThingSpeak Channel and Field
  Description: Writes a value to a channel on ThingSpeak every 20 seconds.
  Hardware: Arduino Uno WiFi Rev2
  Modify the secrets.h file for this project with your network connection and ThingSpe
*/

#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;

Done uploading
Sketch uses 18155 bytes (37%) of program storage space. Maximum is 48640 bytes.
Global variables use 470 bytes (7%) of dynamic memory, leaving 5674 bytes for local va
avrdude: WARNING: invalid value for unused bits in fuse "fuse5", should be set to 1 ac
50 Arduino Uno WiFi Rev2 on COM8
```

```

#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 != "1.0.0.0") {
    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
}

```

This Example uses an Arduino WiFi rev2 board.

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

secrets.h

```

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

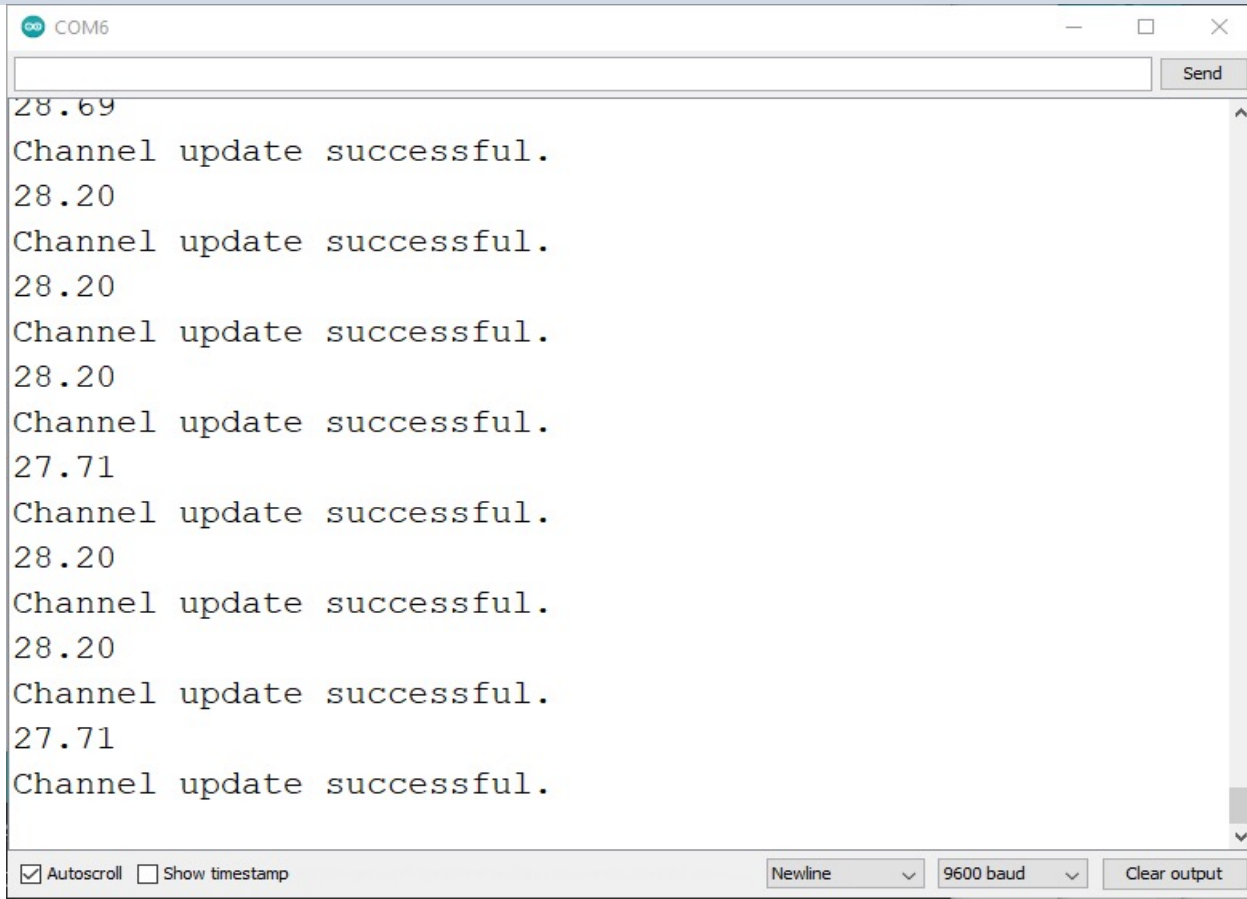
#define SECRET_SSID "MySSID" // replace MySSID with your WiFi network name
#define SECRET_PASS "xxxxxxx" // replace MyPassword with your WiFi password

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

```



# Serial Monitor



The screenshot shows a serial monitor window titled "COM6". The window contains a text area with the following output:

```
28.69  
Channel update successful.  
28.20  
Channel update successful.  
28.20  
Channel update successful.  
28.20  
Channel update successful.  
27.71  
Channel update successful.  
28.20  
Channel update successful.  
28.20  
Channel update successful.  
27.71  
Channel update successful.
```

At the bottom of the window, there are several controls:

- Autoscroll
- Show timestamp
- Newline (dropdown menu)
- 9600 baud (dropdown menu)
- Clear output (button)

# ThingSpeak

We see that the Temperature Data has been successfully written to ThingSpeak

## Field 3 Chart



### TMP36 Temperature



# Updated Code

The Code is the same, but it is now structured into different Functions for better readability

## secrets.h

```
#define SECRET_SSID « xxxxxxxx"  
#define SECRET_PASS « xxxxxxxx"  
  
#define SECRET_CH_ID xxxxxx  
  
#define SECRET_WRITE_APIKEY "xxxxxxx"
```

```
#include "ThingSpeak.h"  
#include <WiFiNINA.h>  
#include "secrets.h"  
  
WiFiClient client;  
int wait = 20000;  
float temperatureValue = 0;  
  
void setup()  
{  
    Serial.begin(9600);  
    CheckWiFi() ;  
    ThingSpeak.begin(client);  
}  
  
void loop()  
{  
    ConnectWiFi() ;  
    ReadTemperature() ;  
    ThingSpeakWrite() ;  
    delay(wait);  
}
```

# CheckWiFi()

```
void CheckWiFi()
{
    // check for the WiFi module:
    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");
    }
}
```

# ConnectWiFi()

```
void ConnectWiFi ()
{
  char ssid[] = SECRET_SSID;
  char pass[] = SECRET_PASS;

  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);
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
}
```

# ReadTemperature()

```
void ReadTemperature ()
{
    int SensorPin = 0;
    float adcValue;
    float voltageValue;

    adcValue = analogRead(SensorPin);
    voltageValue = (adcValue*5)/1023;
    temperatureValue = 100*voltageValue - 50;
    Serial.println(temperatureValue);
}
```

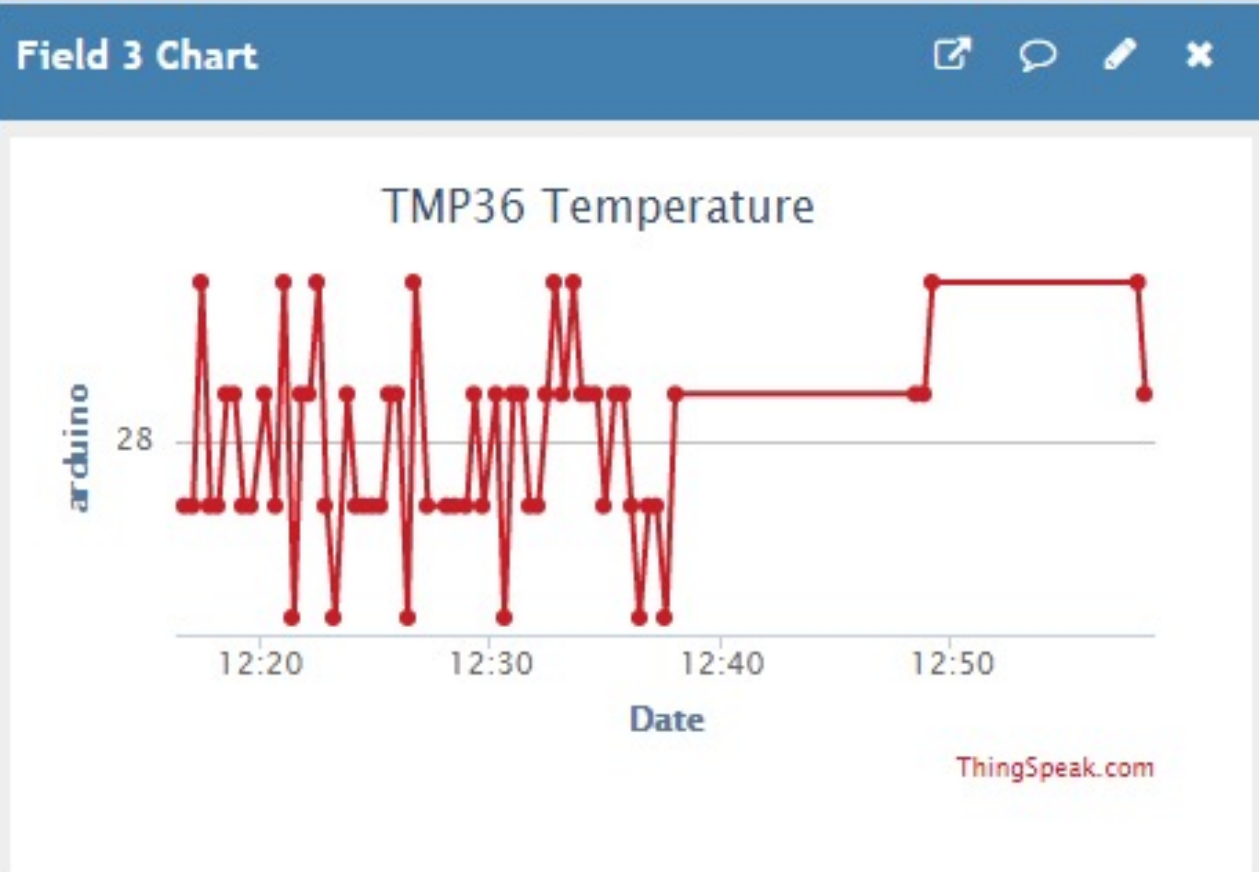
# ThingSpeakWrite()

```
void ThingSpeakWrite()
{
    unsigned long myChannelNumber = SECRET_CH_ID;
    const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
    int channelField = 3;

    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));
    }
}
```

# ThingSpeak

We see that the Temperature Data has been successfully written to ThingSpeak





# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

