Device to Cloud

- Hardware for IoT
- Device anatomy
- Arduino IDE primer
- Sketch Language primer
- Programing and debugging
- Getting data from Sensors
- Temperature
- Light
- Physical sensors
- Ultrasonic
- Sound

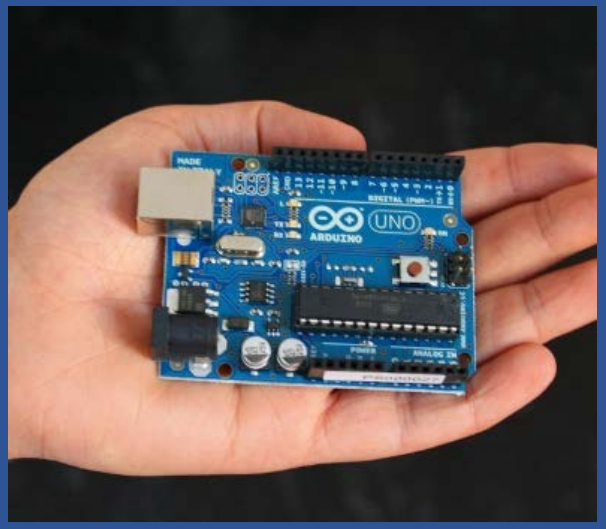
Hardware for IoT

- Prototyping platform
- IoT Starter Kit
- Sensors
- IoT Connectivity
- Loy's IoT Starter Kit

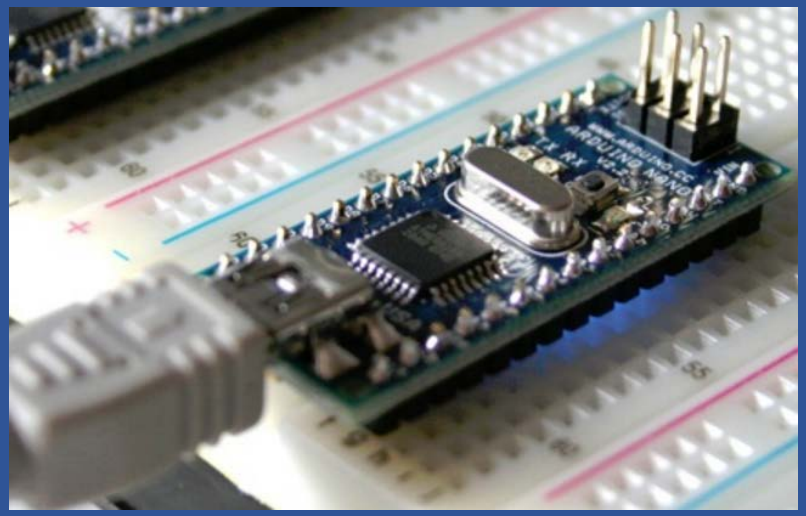
Prototyping platform

- Arduino
- Raspberry Pi
- Intel Edison
- Beagle Board
- NodeMCU (ESP8266)

Arduino



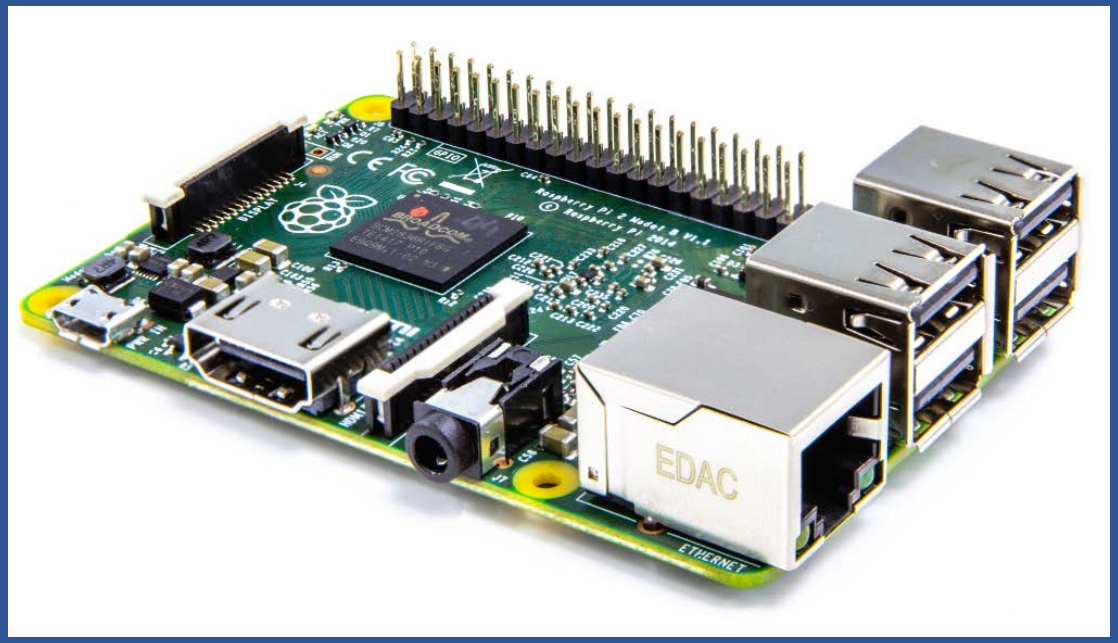
Arduino Uno



Arduino Nano

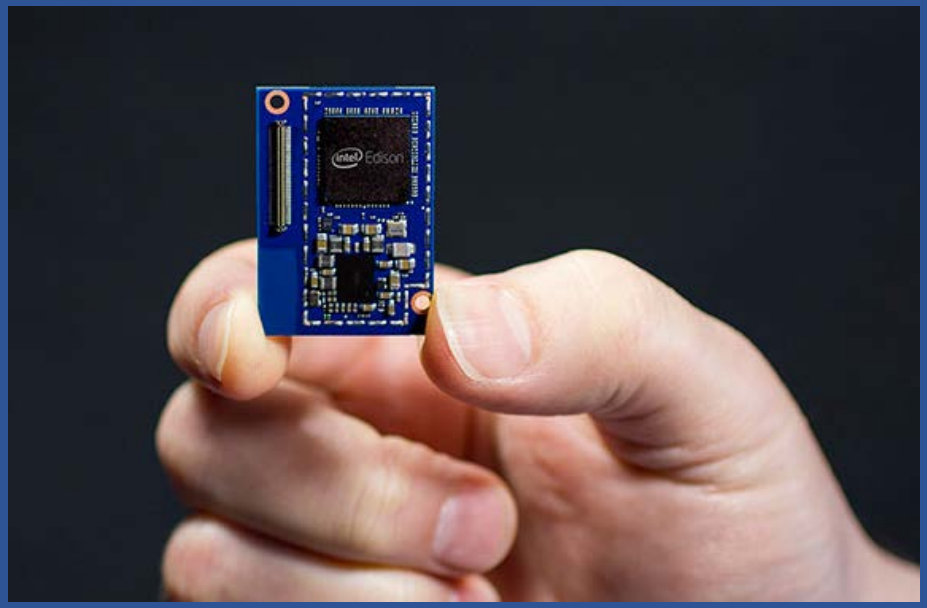
- Micro-controller: ATMEGA
- Open Source
- Developed in Italy
- Language: Arduino sketch

Raspberry Pi



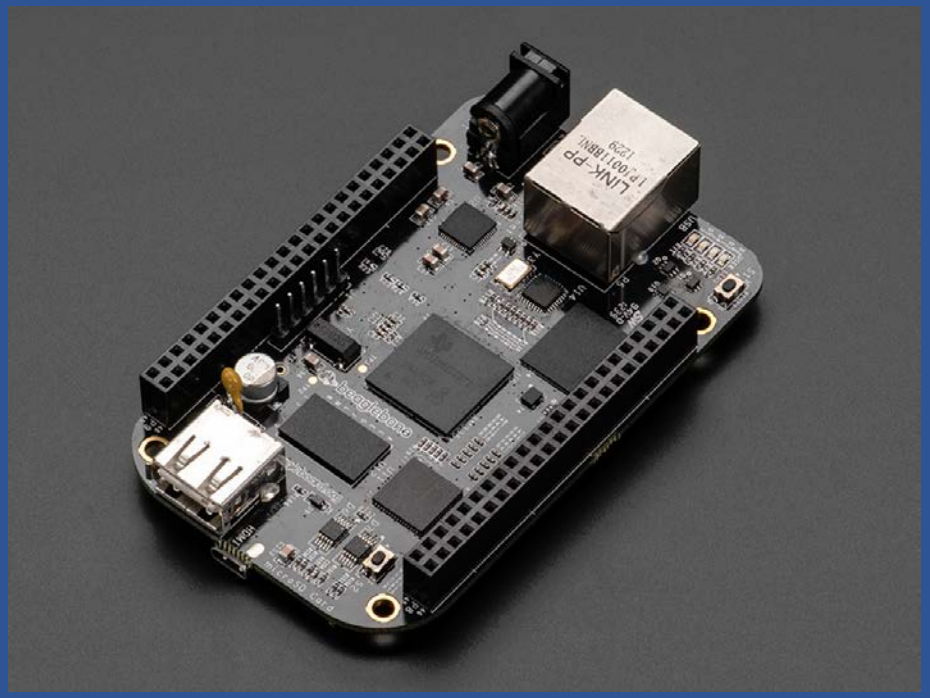
- Micro-controller: ARM Cortex
- Close Source
- Developed in UK
- Language: Java

Intel Edison



- Micro-controller: Intel Atom 2-Core
- Close Source
- Developed in USA
- Language: C / C++

Beagle Board



BeagleBone Black

- Micro-controller: ARM Cortex
- Open Source
- Developed in USA
- Language: Java

IoT Starter Kit

Microsoft Azure

- **Electronic components**
  - **1x** Photo Cell
  - **2x** Breadboard Trim Potentiometer
  - **5x** 10K 5% 1/4W Resistor
  - **5x** 560 ohm 5% 1/4W Resistor
  - **2x** Diffused 10mm Blue LED
  - **1x** Electrolytic Capacitor - 1.0uF
  - **2x** Diffused 10mm Red LED
  - **2x** Diffused 10mm Green LED
  - **3x** 12mm Tactile Switches

Adafruit Feather board: Raspberry Pi

Grove Starter Kit for Arduino

Parts list

1xBase Shield
1xGrove - LCD RGB Backlight
1xGrove - Smart Relay
1xGrove - Buzzer
1xGrove - Sound Sensor
1xGrove - Touch Sensor
1xGrove - Rotary Angle Sensor
1xGrove - Temperature Sensor
1xGrove - LED
1xGrove - Light Sensor
1xGrove – Button
1xDIP LED Blue-Blue
1xDIP LED Green-Green
1xDIP LED Red-Red
1xMini Servo
10xGrove Cables
1x9V to Barrel Jack Adapter
1xGrove starter kit Manual
1xGreen Plastic Box

amazon web services™

Smart Living Starter Kit for Intel Edison

Sensors & Actuators

- Temperature sensor
- Light sensor
- Motion sensor
- Infrared modules
- Vibration motor
- LED
- LED bar
- Rotary knob
- Push button

SmartLiving Maker

BeagleBone Green IOT for AWS

amazon web services™

Package including:
BeagleBone Green x1
Grove - I2C ADC x1
Grove - I2C Hub x1
Grove - Relayx1
Grove - Buzzer x1
Grove - OLED Display 0.96" x1
Grove - Button x1
Grove - Temperature Sensor x1
Grove - Sound Sensor x1
Grove - 3-Axis Digital Accelerometer(±16g) x1
Grove - Chainable RGB LED x1
Micro USB Cable - 48cm x1

ARM mbed Mbed NXP

IoT Device Platform

Ethernet IoT Starter Kit

- NXP K64F Processor
- mbed application shield
- IBM IoT Client pre-loaded

FRDM-K64F

- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Ethernet, USB Crystal-less, SI

FRDM-KL46Z

- Cortex-M0+, 48MHz
- 256KB Flash, 32KB RAM
- USB OTG

NXP LPC800-MAX

- Cortex-M0+
- 16KB Flash, 4KB RAM

Extract (30 MCU platforms for NXP)

full list

Focus for current mbed OS release

FRDM-STBC-AGM01: 9-Axis IMU

FRDM-FXS-MULTI2-B

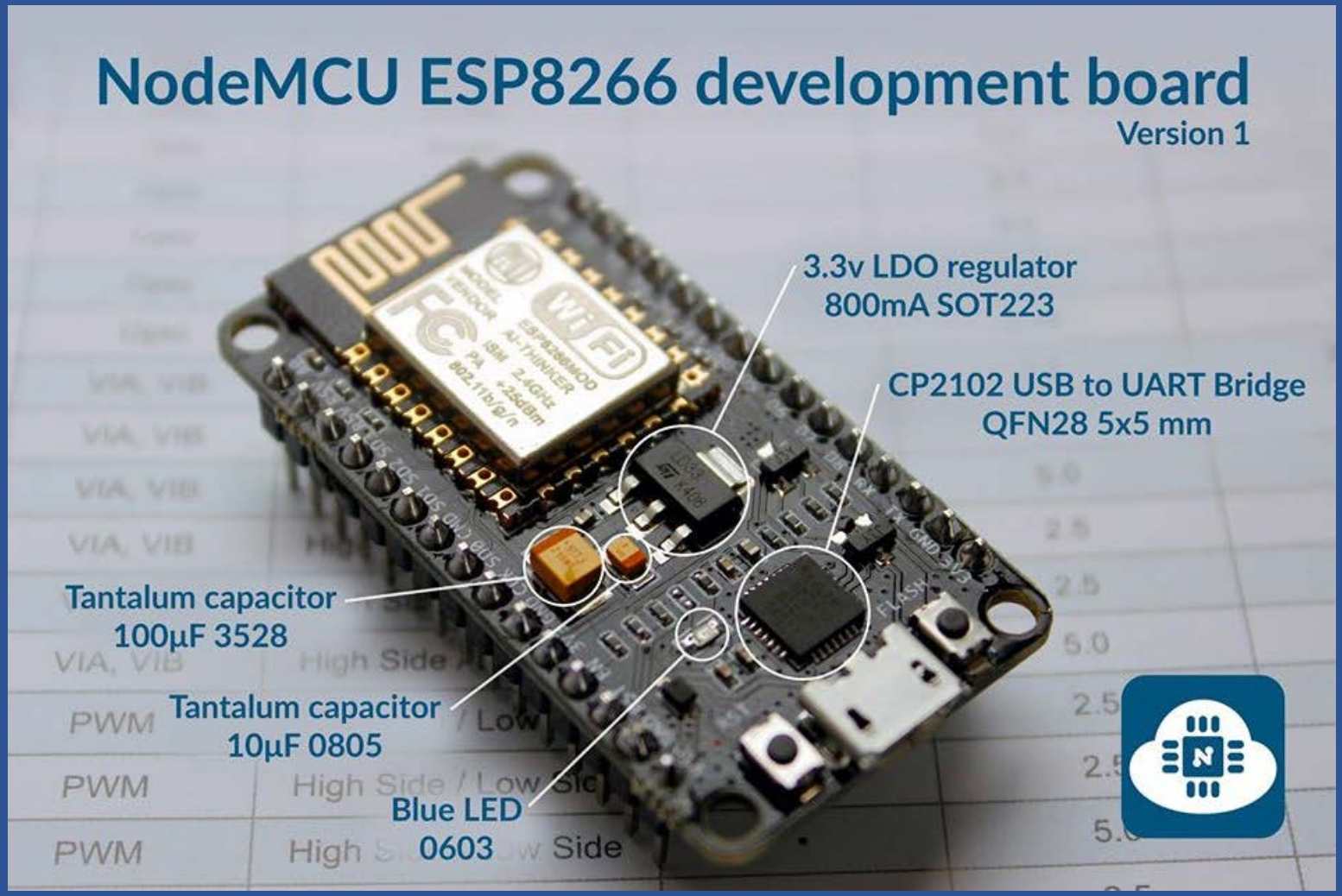
FRDMSTBC-A8471

FRDMSTBC-A8491

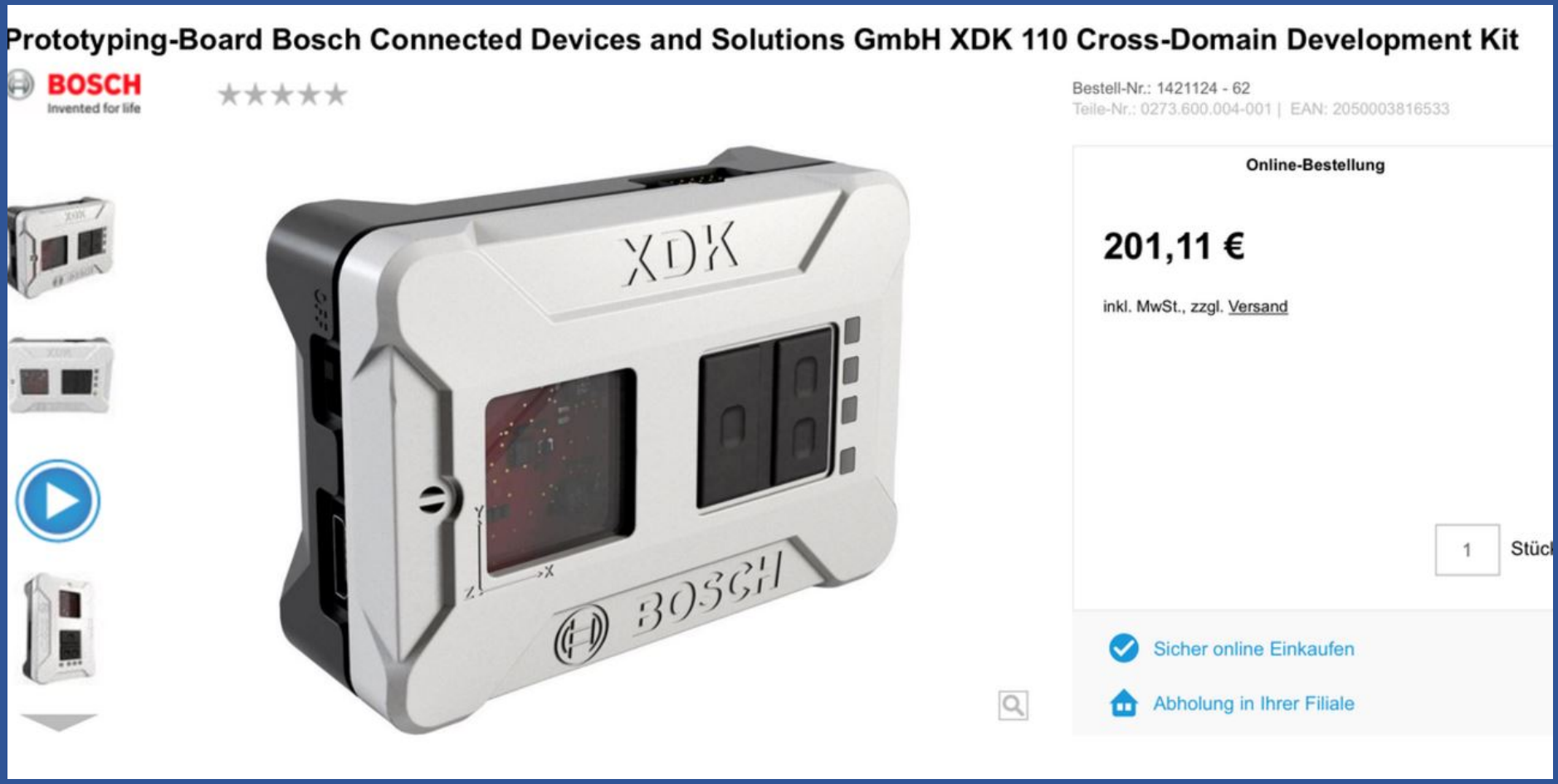
FRDM-CR20A 2.4GHz 802.15.4 Wireless Transceiver

JN5179

NodeMCU


NodeMCU ESP8266 development board
Version 1
3.3v LDO regulator
800mA SOT223
CP2102 USB to UART Bridge
QFN28 5x5 mm
Tantalum capacitor
100µF 3528
Tantalum capacitor
10µF 0805
Blue LED
0603

BOSCH XDK


Prototyping-Board Bosch Connected Devices and Solutions GmbH XDK 110 Cross-Domain Development Kit
BOSCH
Invented for life
Bestell-Nr.: 1421124 - 62
Teile-Nr.: 0273.600.004-001 | EAN: 2050003816533
Online-Bestellung
201,11 €
inkl. MwSt., zzgl. Versand
1
Stück
Sicher online Einkaufen
Abholung in Ihrer Filiale
XDK
BOSCH

More On IoT Hardware

- Microsoft Azure IoT Starter Kits

https://azure.microsoft.com/en-us/develop/iot/starter-kits/

- EBV Elektronik NXP IoT Berlin Presentations

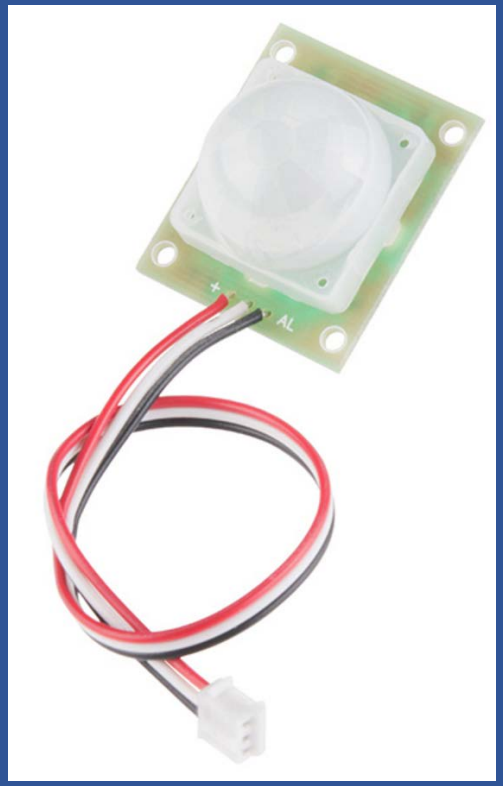
https://www.dropbox.com/sh/iqo7d7906khrt5v/AAC0bZy2IY6Ekp8QUQ8zo0bla?dl=0

Sensors

- Acoustic, sound, vibration
- Automotive, transportation
- Chemical
- Electric current, electric potential, magnetic, radio
- Flow, fluid velocity
- Ionizing radiation, subatomic particles
- Navigation instruments
- Position, angle, displacement, distance, speed, acceleration
- Optical, light, imaging, photon
- Pressure
- Force, density, level
- Thermal, heat, temperature
- Proximity, presence

Motion Sensor

https://www.sparkfun.com/products/13285



Description: This is a simple to use motion sensor. Power it up and wait 1-2 seconds for the sensor to get a snapshot of the still room. If anything moves after that period, the 'alarm' pin will go low.

This unit works great from 5 to 12V (datasheet shows 12V). You can also install a jumper wire past the 5V regulator on board to make this unit work at 3.3V. Sensor uses 1.6mA@3.3V.

Altitude/Pressure Sensor

https://www.sparkfun.com/products/13754

Description: The MPL3115A2 is a MEMS pressure sensor that provides Altitude data to within 30cm (with oversampling enabled).

The sensor outputs are digitized by a high resolution 24-bit ADC and transmitted over I2C, meaning it's easy to interface with most controllers.

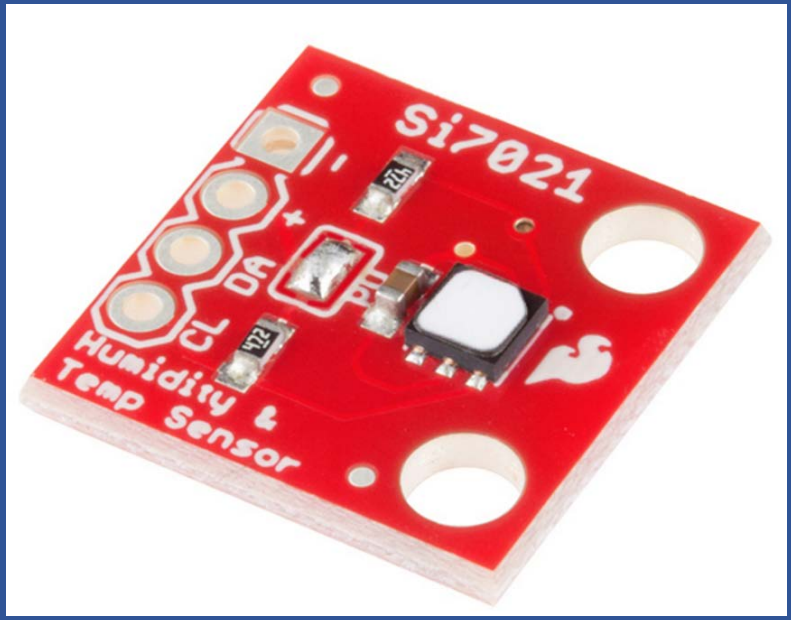
Pressure output can be resolved with output in fractions of a Pascal, and Altitude can be resolved in fractions of a meter. The device also provides 12-bit temperature measurements in degrees Celsius.

Photocell

Description: The photocell will vary its resistance based on how much light it's exposed to. Will vary from 1kΩ in the light to 10kΩ in the dark.

Humidity and Temperature Sensor Breakout- Si7021

https://www.sparkfun.com/products/13763



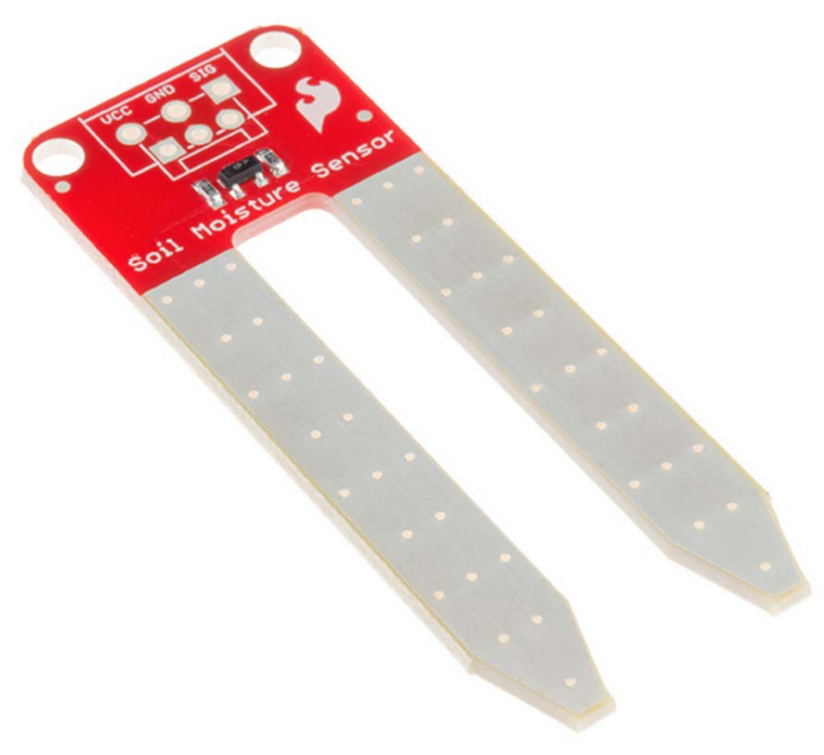
Description: The Si7021 is a low-cost, easy-to-use, highly accurate, digital humidity and temperature sensor.

This sensor is ideal for environmental sensing and data logging and perfect for build a weather stations or humidor control system.

All you need are two lines for I2C communication, and you'll have relative humidity readings and very accurate temperature readings as a bonus!

Soil Moisture Sensor

https://www.sparkfun.com/products/13322

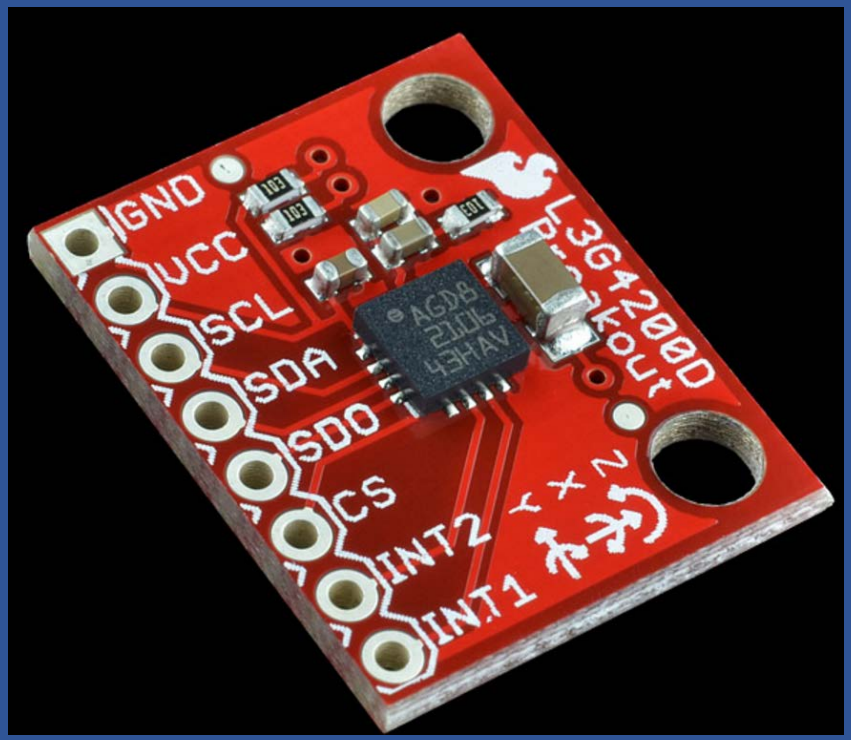


Description: The two large exposed pads function as probes for the sensor, together acting as a variable resistor.

The more water that is in the soil means the better the conductivity between the pads will be and will result in a lower resistance, and a higher SIG out.

Tri-Axis Gyro

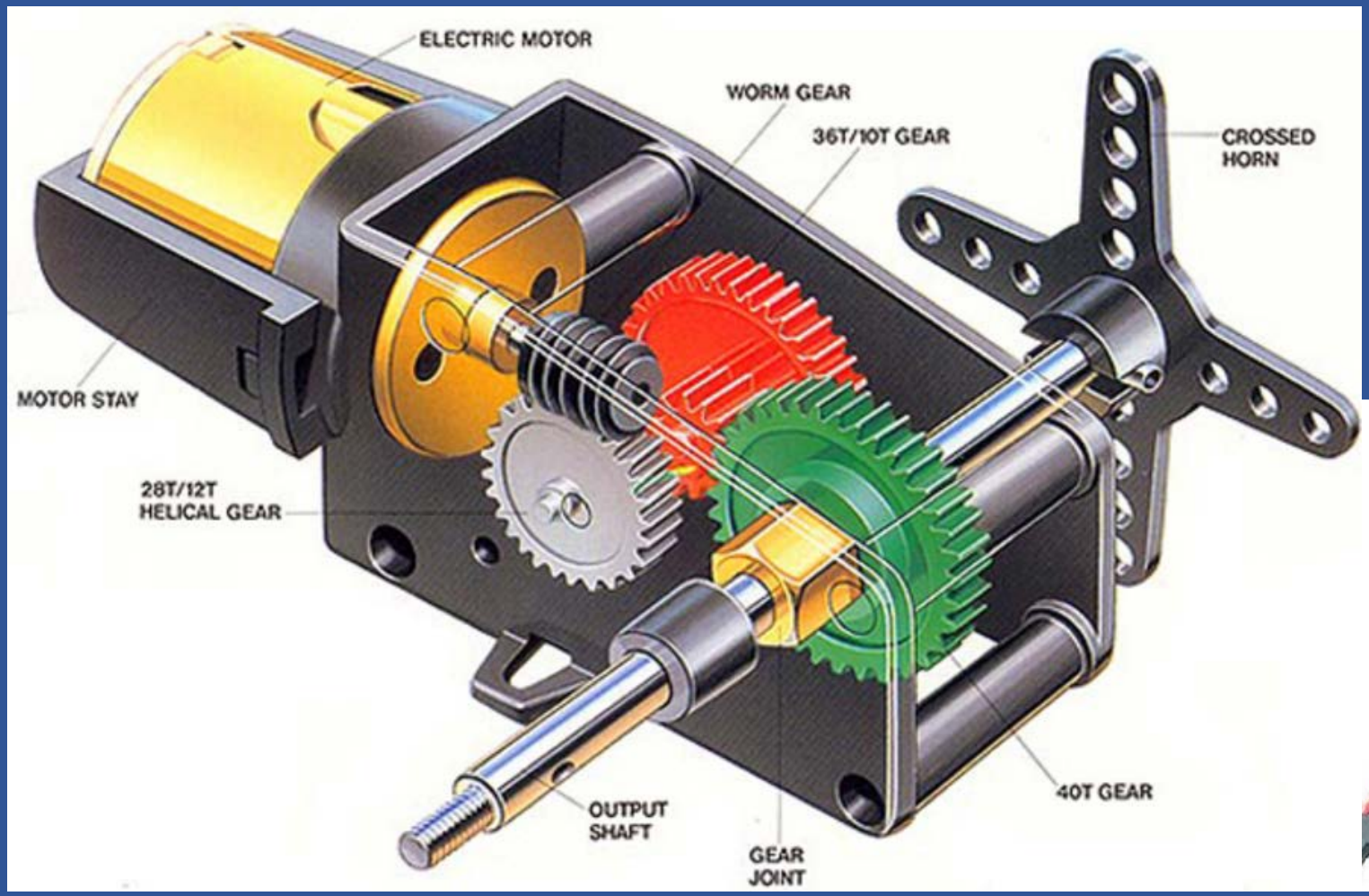
https://www.sparkfun.com/products/10612



Description: Angular velocity is the change in rotational angle per unit of time. Angular velocity is generally expressed in deg/s (degrees per second).

These work great in gaming and virtual reality input devices, GPS navigation systems and robotics.

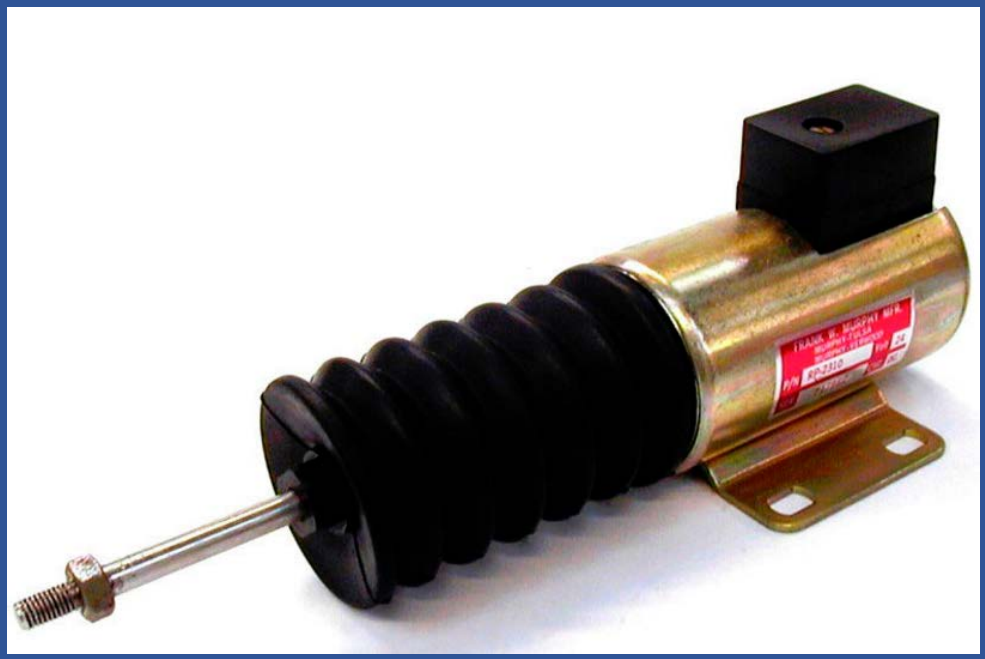
Servo Motor


ELECTRIC MOTOR
WORM GEAR
36T/10T GEAR
CROSSED HORN
MOTOR STAY
28T/12T HELICAL GEAR
OUTPUT SHAFT
GEAR JOINT
40T GEAR

Robot Arm

DC gear motor

Solenoid



IoT Connectivity

- PAN: Personal Area Network
- LAN: Local Area Network
- NAN: Neighbourhood Area Network
- WAN: Wide Area Network

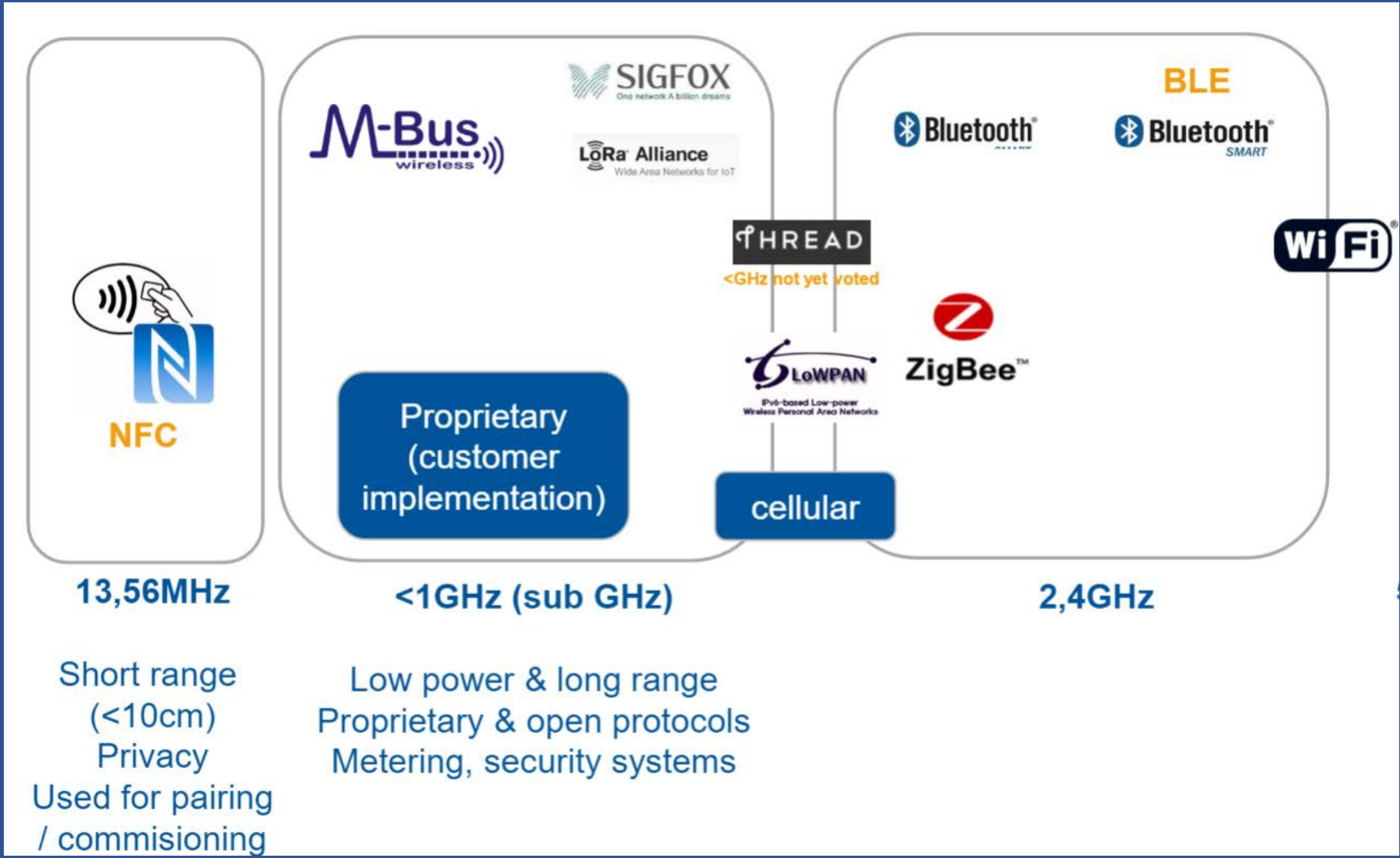
- Bluetooth low energy (BLE)
- applications in the healthcare, fitness, beacons, security, and home entertainment industries.
- Reduced power consumption and cost while maintaining a similar communication range.
- Mobile operating systems including iOS, Android, Windows Phone and BlackBerry, as well as OS X, Linux, and Windows 8, natively support Bluetooth Smart.
- 2018 more than 90 percent of Bluetooth-enabled smartphones will support Bluetooth Smart.

- ZigBee is an IEEE 802.15.4-based specification
- Used to create personal area networks with small, low-power digital radios.
- Intended to be simpler and less expensive than Bluetooth or Wi-Fi.
- Short-range low-rate wireless data transfer.
- Applications include wireless light switches, traffic management systems, and other consumer and industrial equipment

- 6LoWPAN is an acronym of IPv6 over Low power Wireless Personal Area Networks.
- Concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices," and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.
- Allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks.

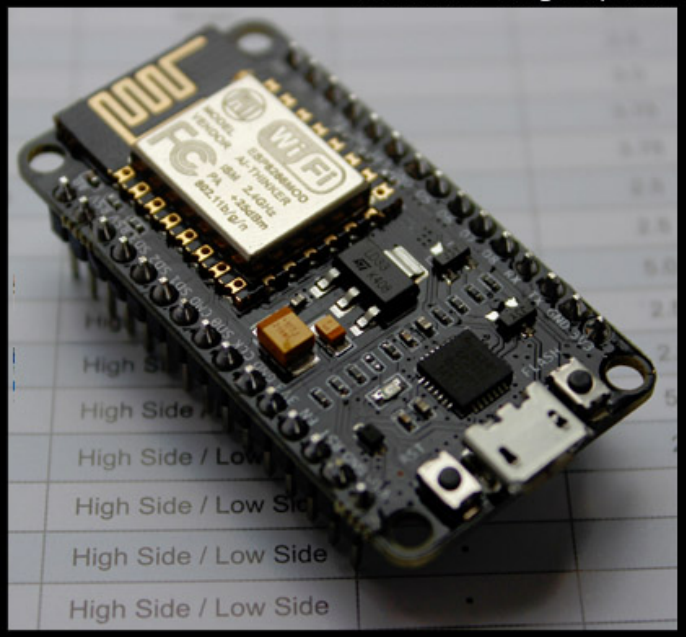
- French company
- Wireless networks to connect low-energy objects
- Electricity meters, smartwatches, and washing machines
- Continuously on and emitting small amounts of data.
- $1 per device per year
- First and only company providing global cellular connectivity for the Internet of Things"
- Infrastructure is "completely independent of existing networks, such as telecommunications networks."

Wireless connectivity for IoT

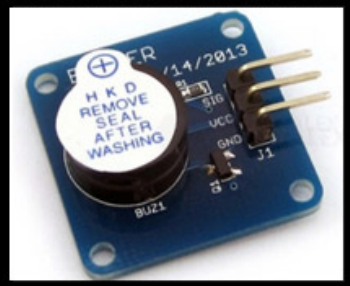


LOY IoT STARTER KIT™

Internet of Things experimental kit for Microsoft Azure IoT suite training course by Laploy

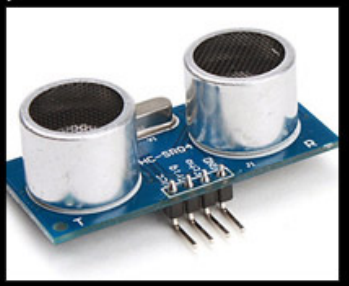


NodeMCU ESP8266



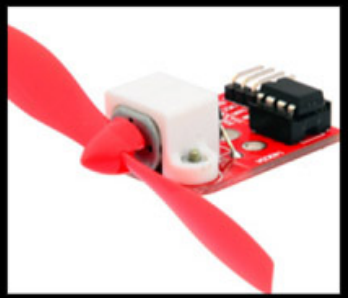
BUZZER

RELAY

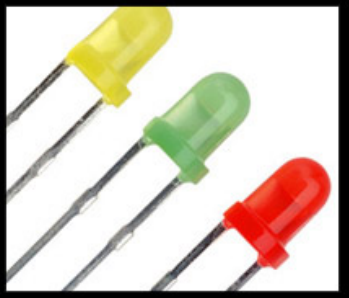


ULTRASONIC SENSOR

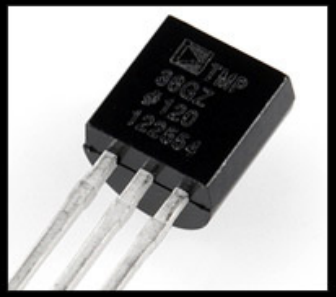
ANGLE SENSOR



FAN

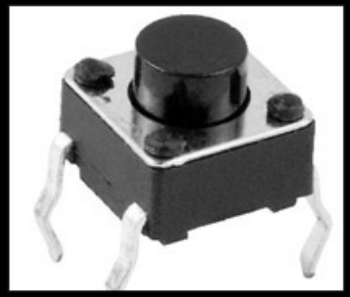


LED

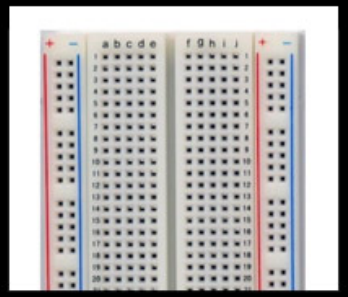


TEMPERATURE

LIGHT SENSOR



SWITCH

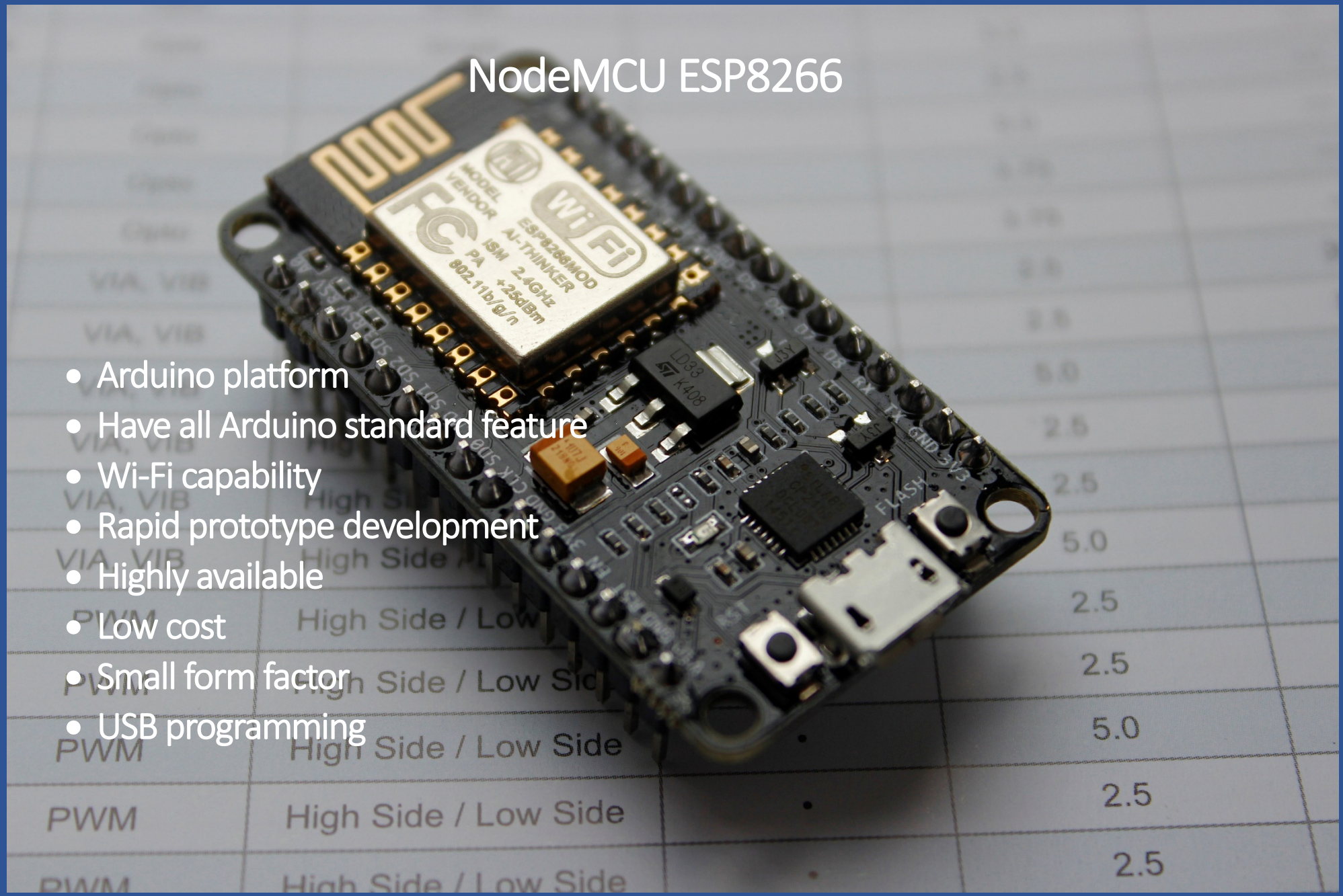


BREADBOARD

USB CABLE

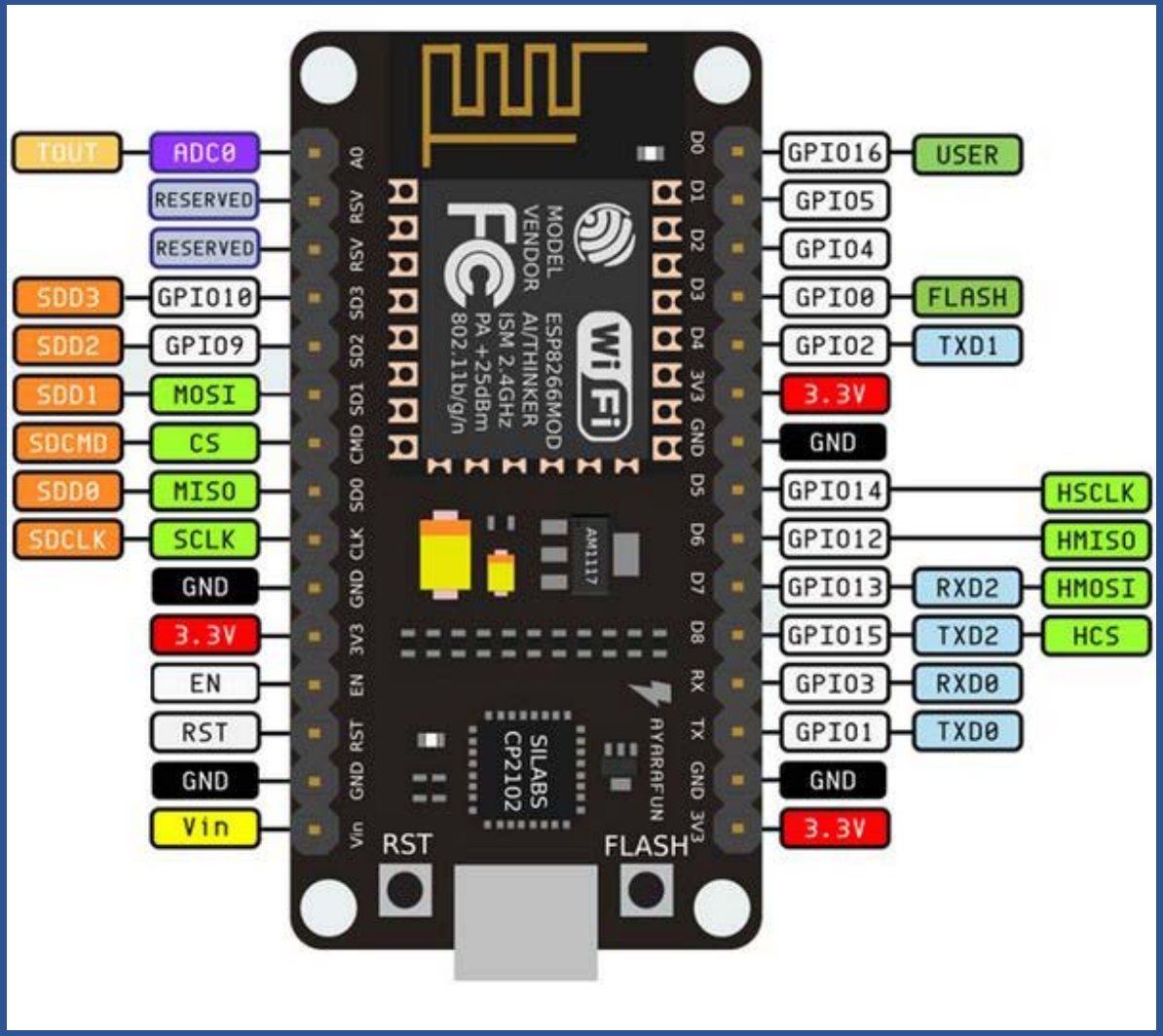
The picture only serves general information purposes; it may be subject to change at any time without prior notice.

NodeMCU ESP8266

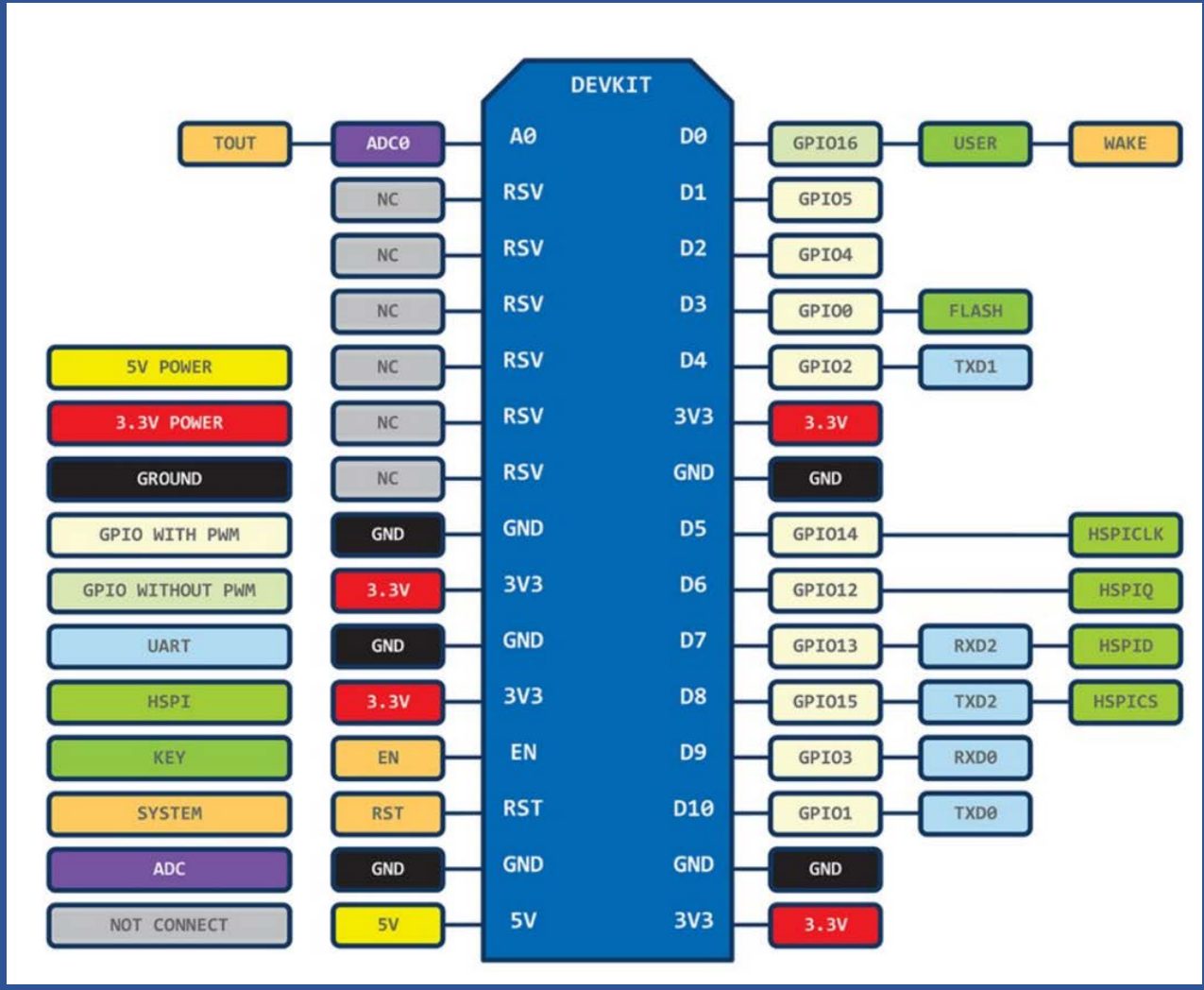


- Arduino platform
- Have all Arduino standard feature
- Wi-Fi capability
- Rapid prototype development
- Highly available
- Low cost
- Small form factor
- USB programming

Device anatomy



Pin description



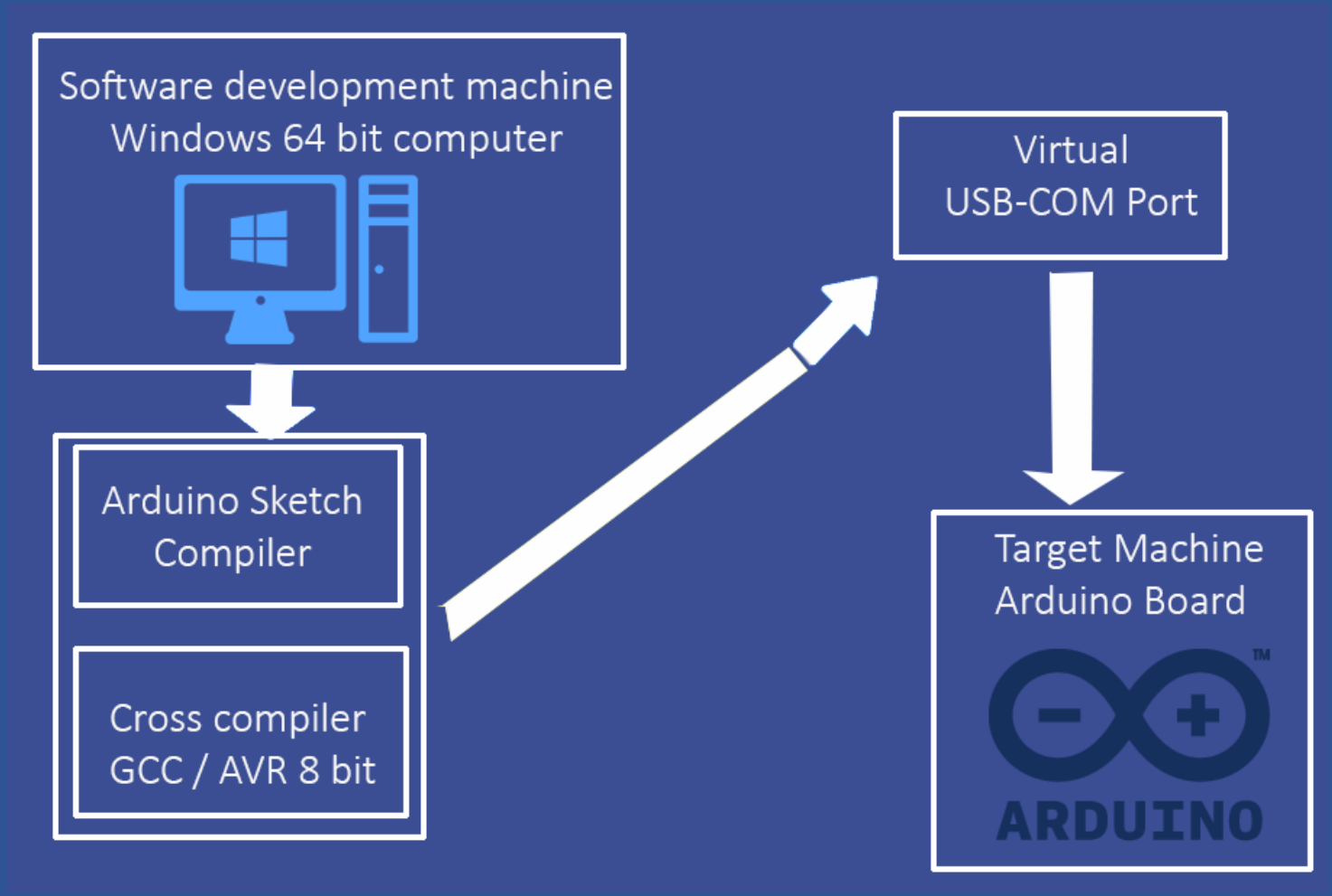
32-bit Tensilica MCU

- Tensilica L106 32-bit micro controller (MCU)
- Extra low power
- 32-bit RSIC,
- Clock speed of 160 MHz.
- Real Time Operation System (RTOS)
- Wi-Fi stack

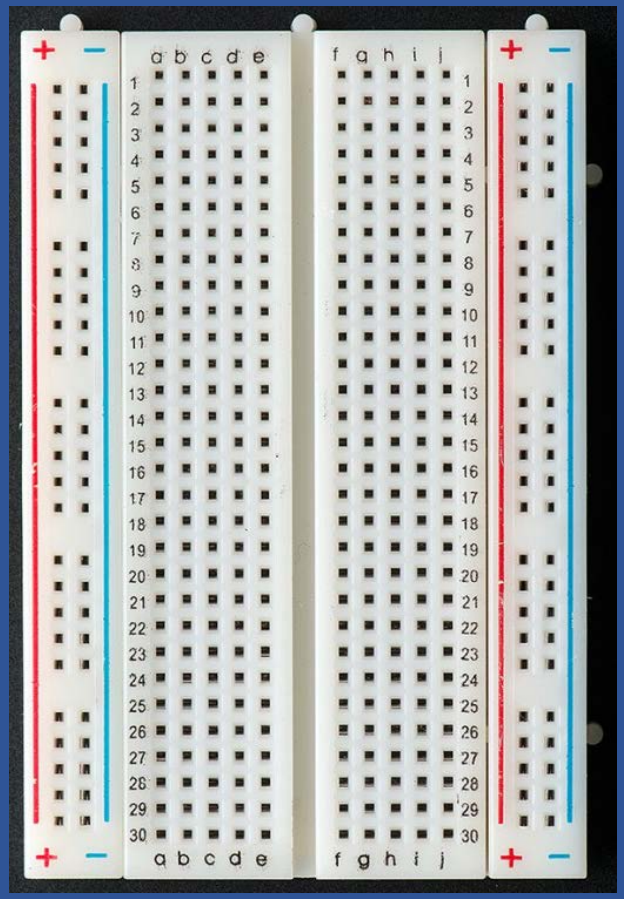
Peripheral features

- 32-bit RISC CPU: Tensilica Xtensa LX106 running at 80 MHz*
- 64 KB of instruction RAM, 96 KB of data RAM
- External QSPI flash- 512 KB to 4 MB* (up to 16 MB is supported)
- IEEE 802.11 b/g/n Wi-Fi
- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI, I^2C,
- I^2S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 1 10-bit ADC

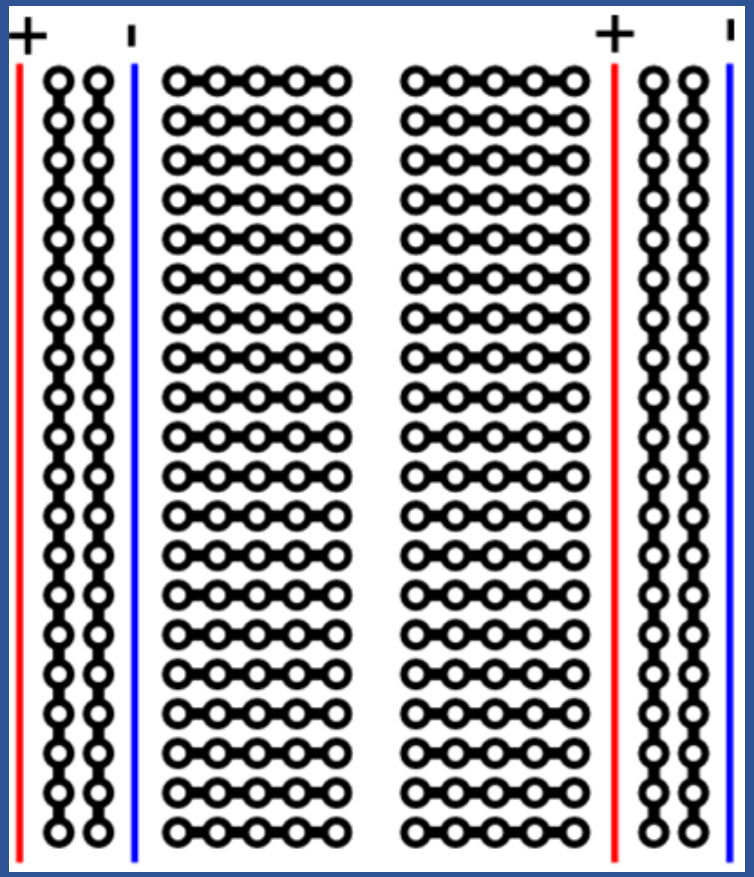
PC – Arduino Cross Compile programming model



Breadboard

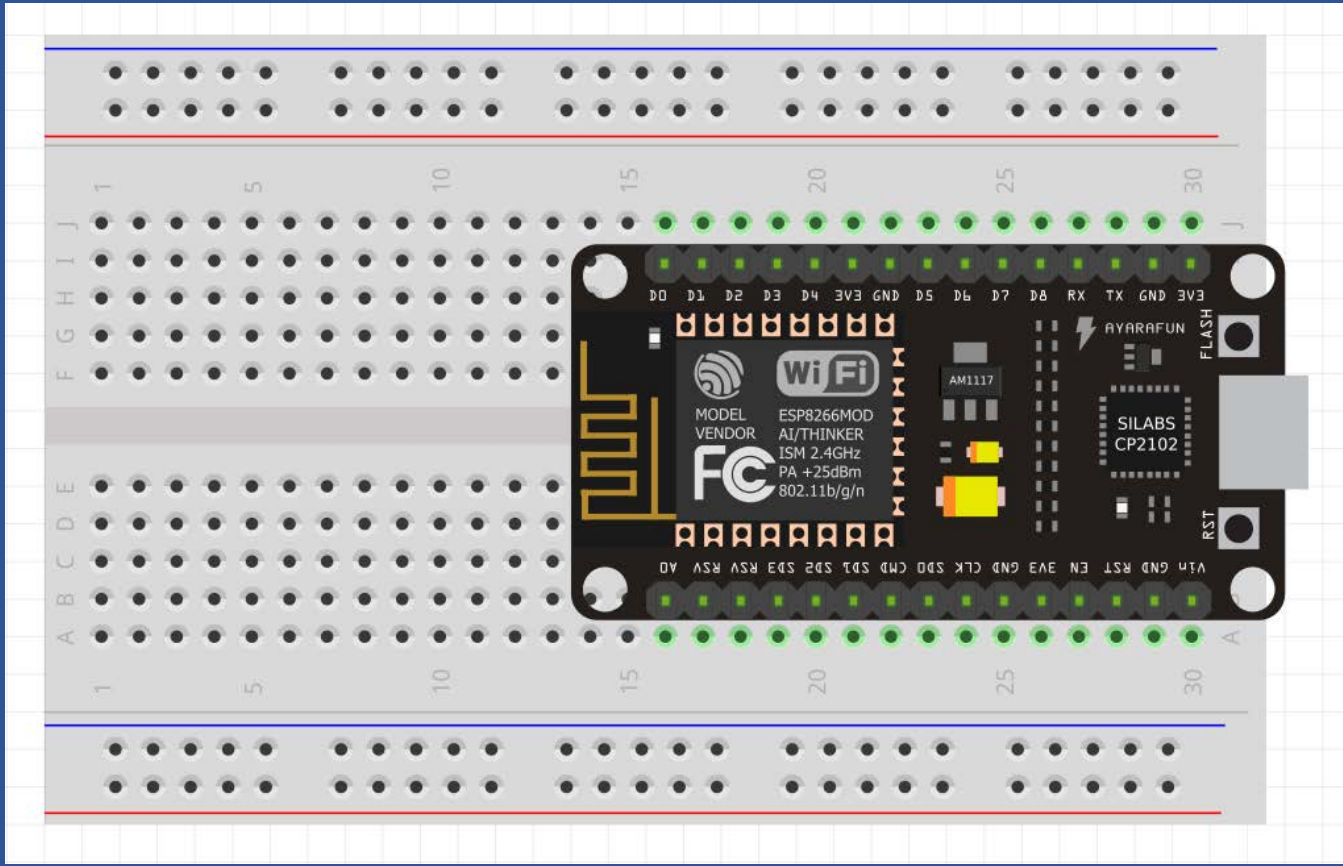


Actual

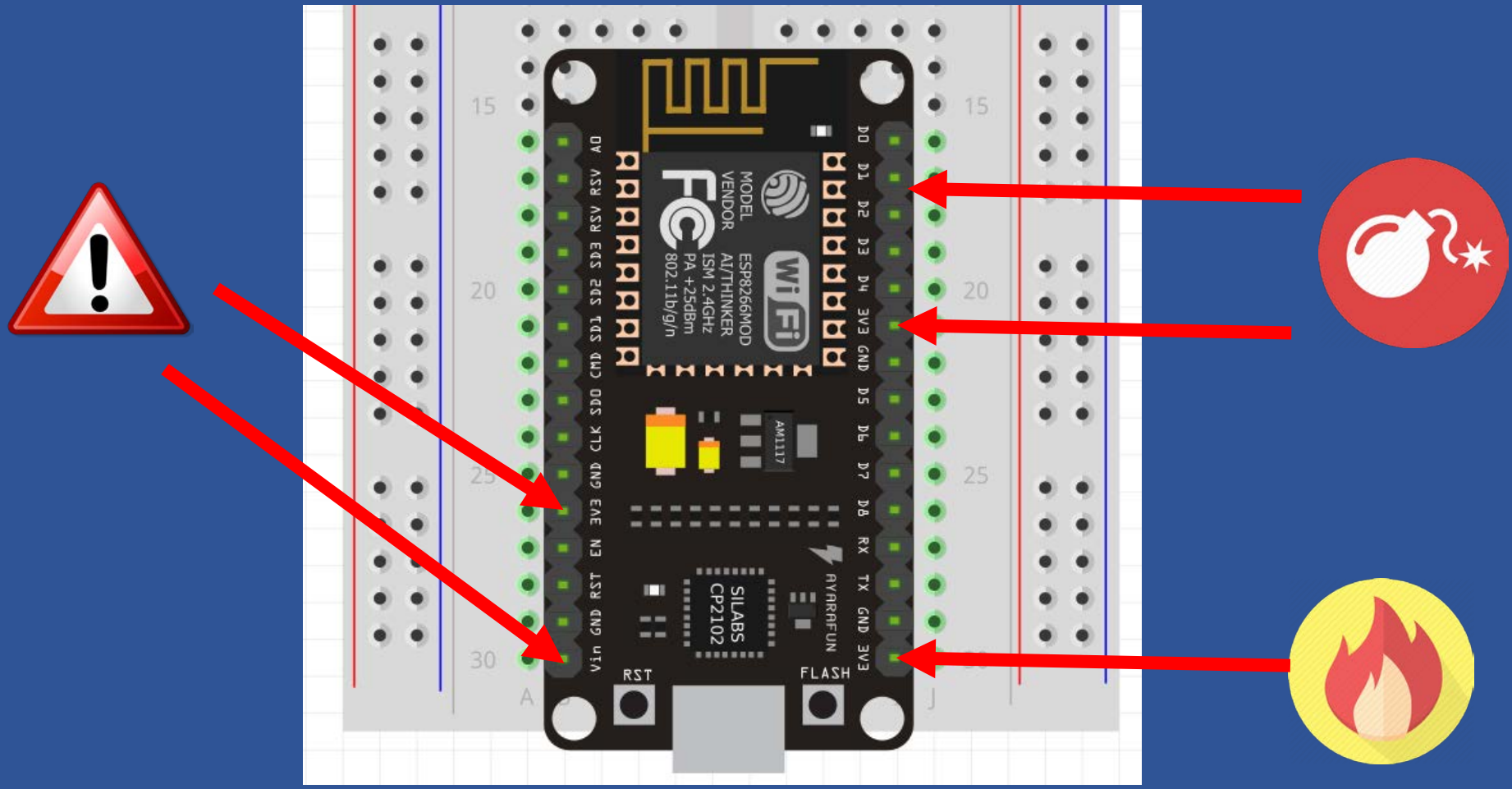


Schematic

NodeMCU mounted on Breadboard



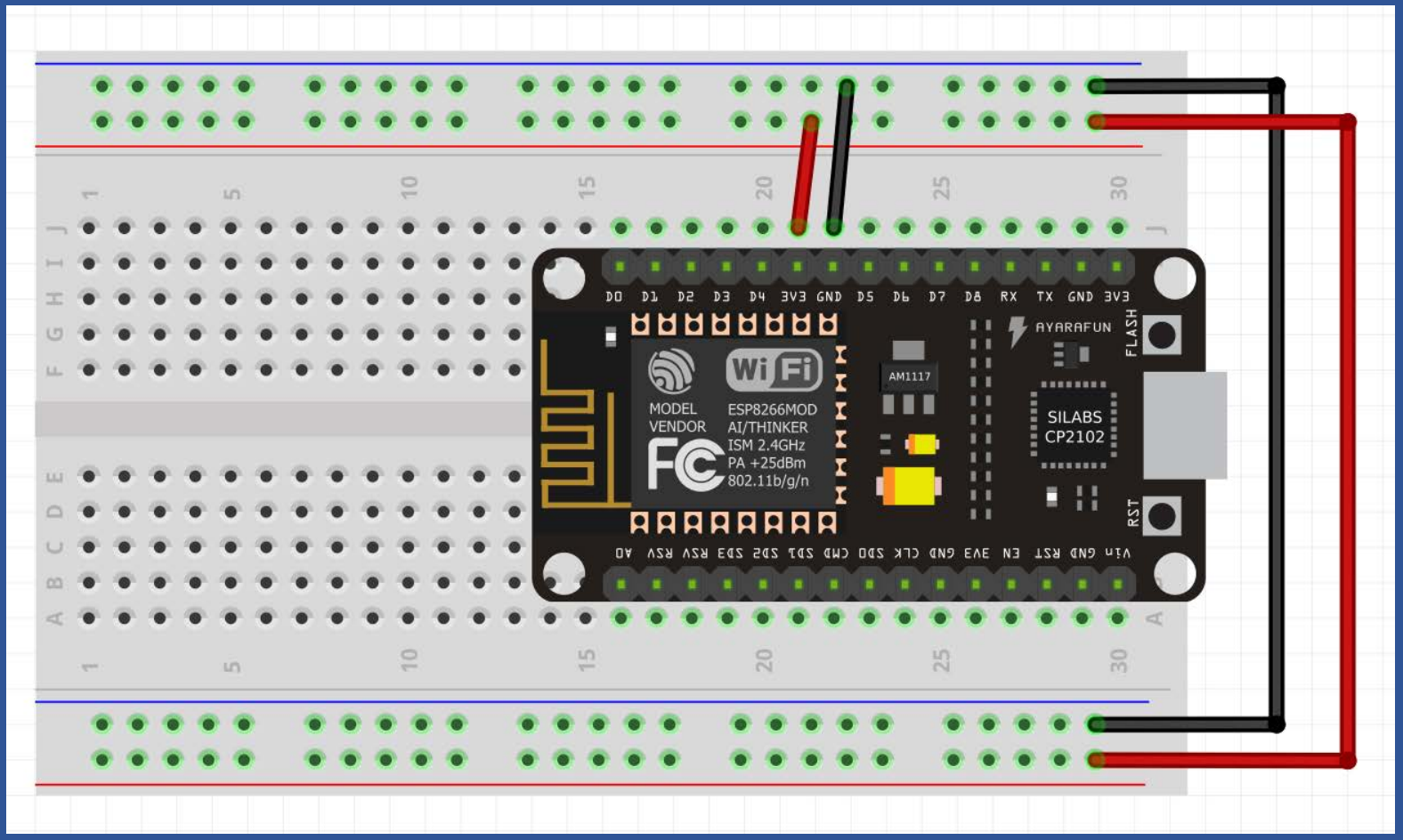
Hot Pins


15
20
25
30
A
J
MODEL
VENDOR
ESP8266MOD
AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
AM1117
SILABS
CP2102
AYARAFUN
RST
FLASH
Vin GND RST EN 3V3 GND CLK SDO CMD SD1 SD2 SD3 RSV RSV A0
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3

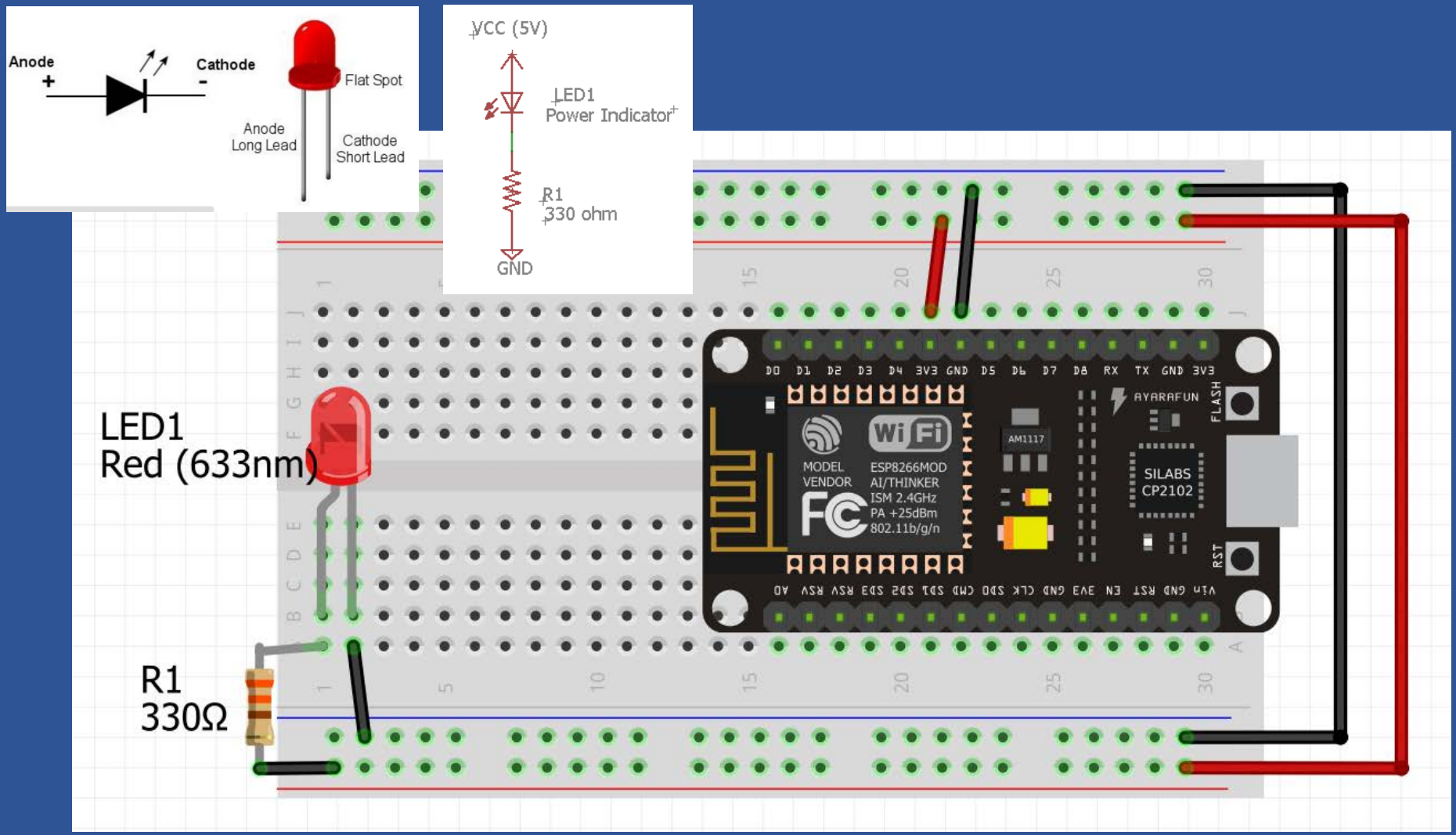
Safety Instructions

- Remove USB when not in used
- Remove USB when modify circuit
- Remove USB when hot
- Remove USB when smoke
- Do not connect hot pins to GND
- Check, re-check and cross-check before plug USB
- Immediately remove USB if no light

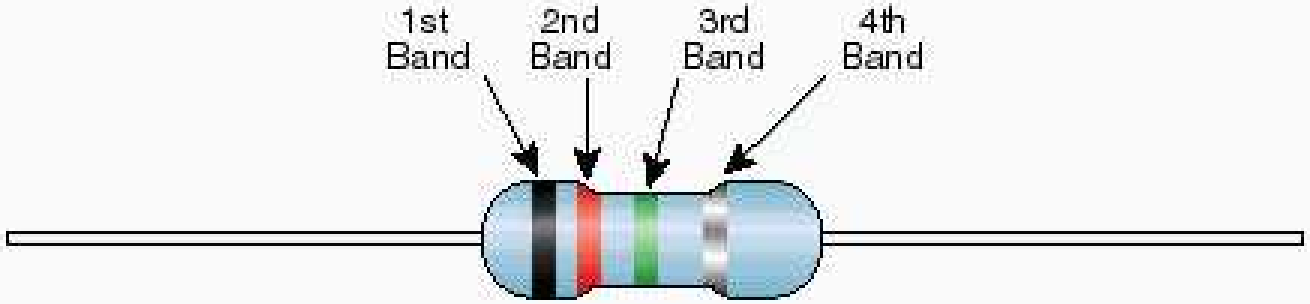
Powering the breadboard



Power indicator

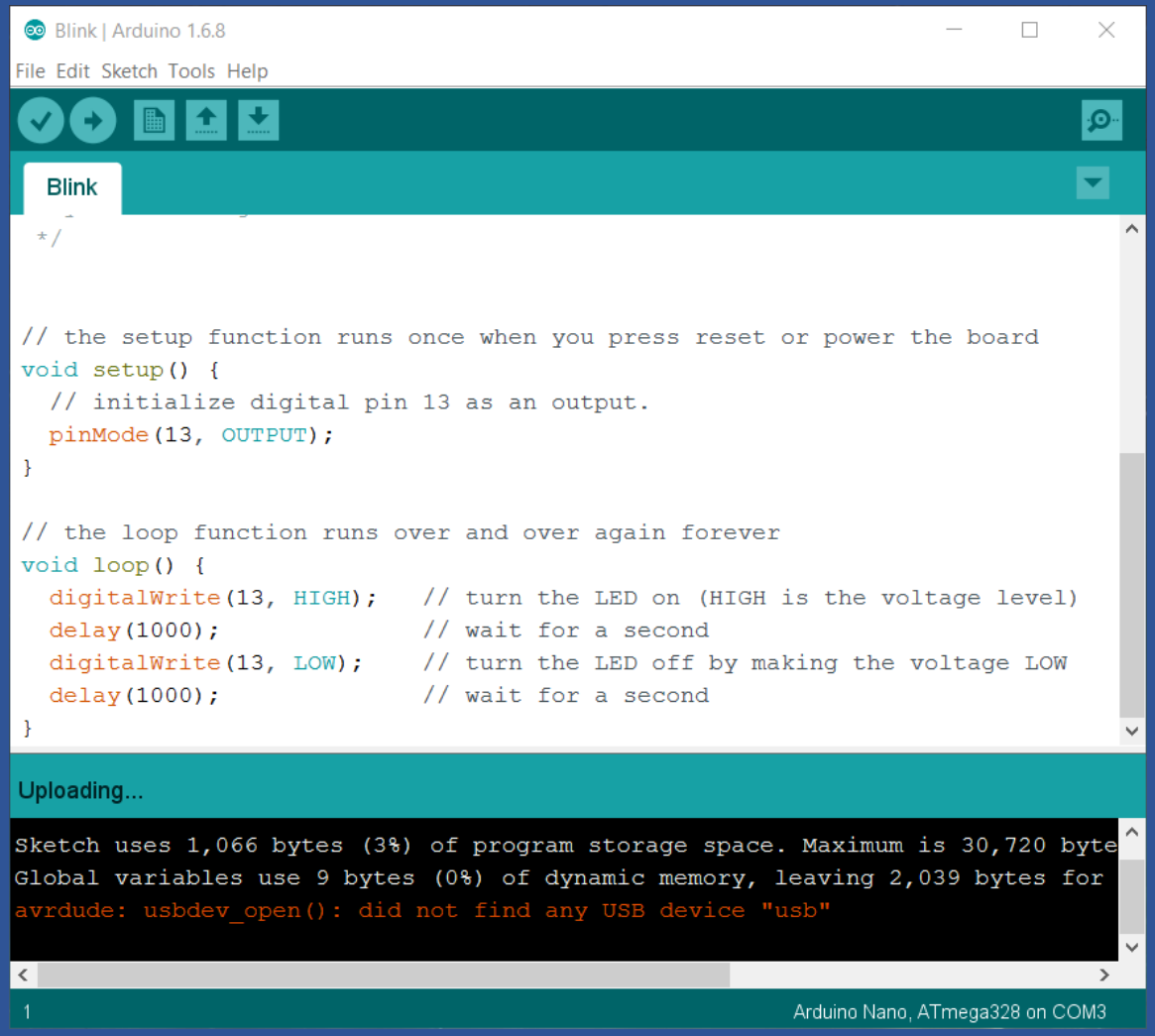


Standard EIA Color Code Table 4 Band: ±2%, ±5%, and ±10%



Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	±2%
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	±5%
Silver			10^{-2}	±10%

Arduino IDE primer



More on Arduino IDE

- Arduino Software (IDE)

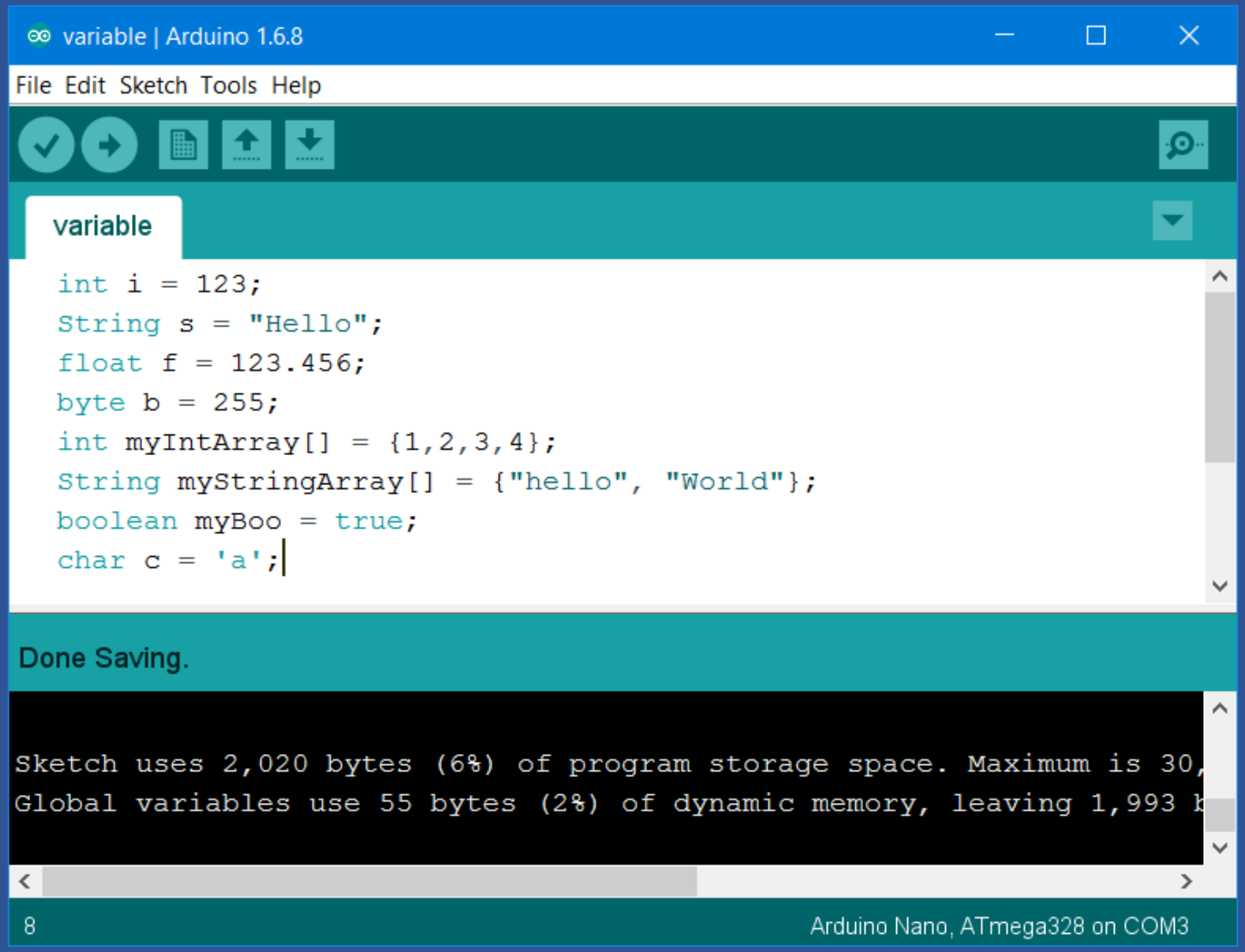
https://www.arduino.cc/en/Guide/Environment

Sketch Language primer

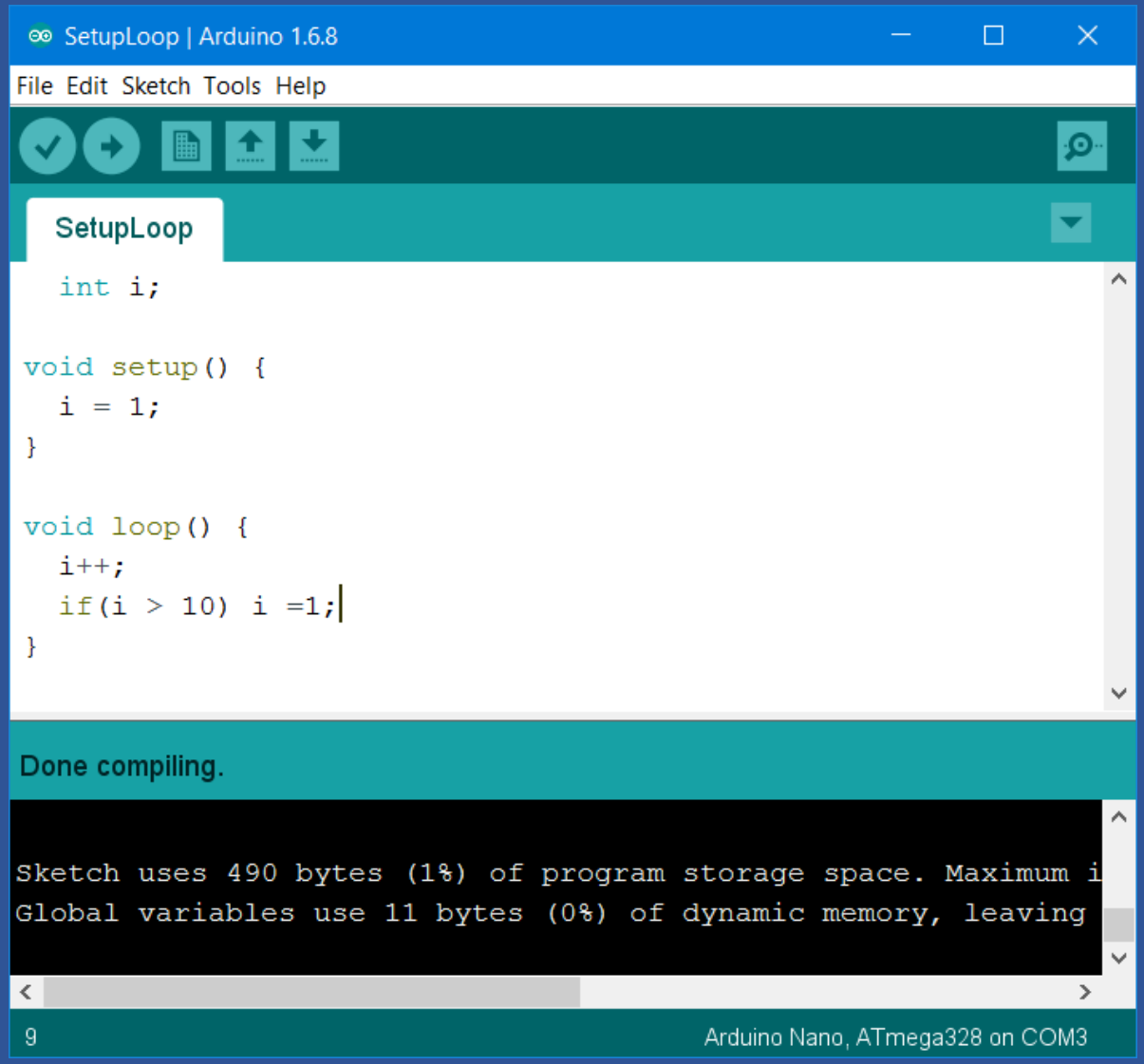
Data type

Datatype	RAM usage
• void keyword	N/A
• boolean	1 byte
• char	1 byte
• unsigned char	1 byte
• int	2 byte
• unsigned int	2 byte
• word	2 byte
• long	4 byte
• unsigned long	4 byte
• float	4 byte
• double	4 byte
• string	1 byte + x
• array	1 byte + x
• enum	N/A
• struct	N/A
• pointer	N/A

Variable Declaration



Setup and Loop



Blinky Lab

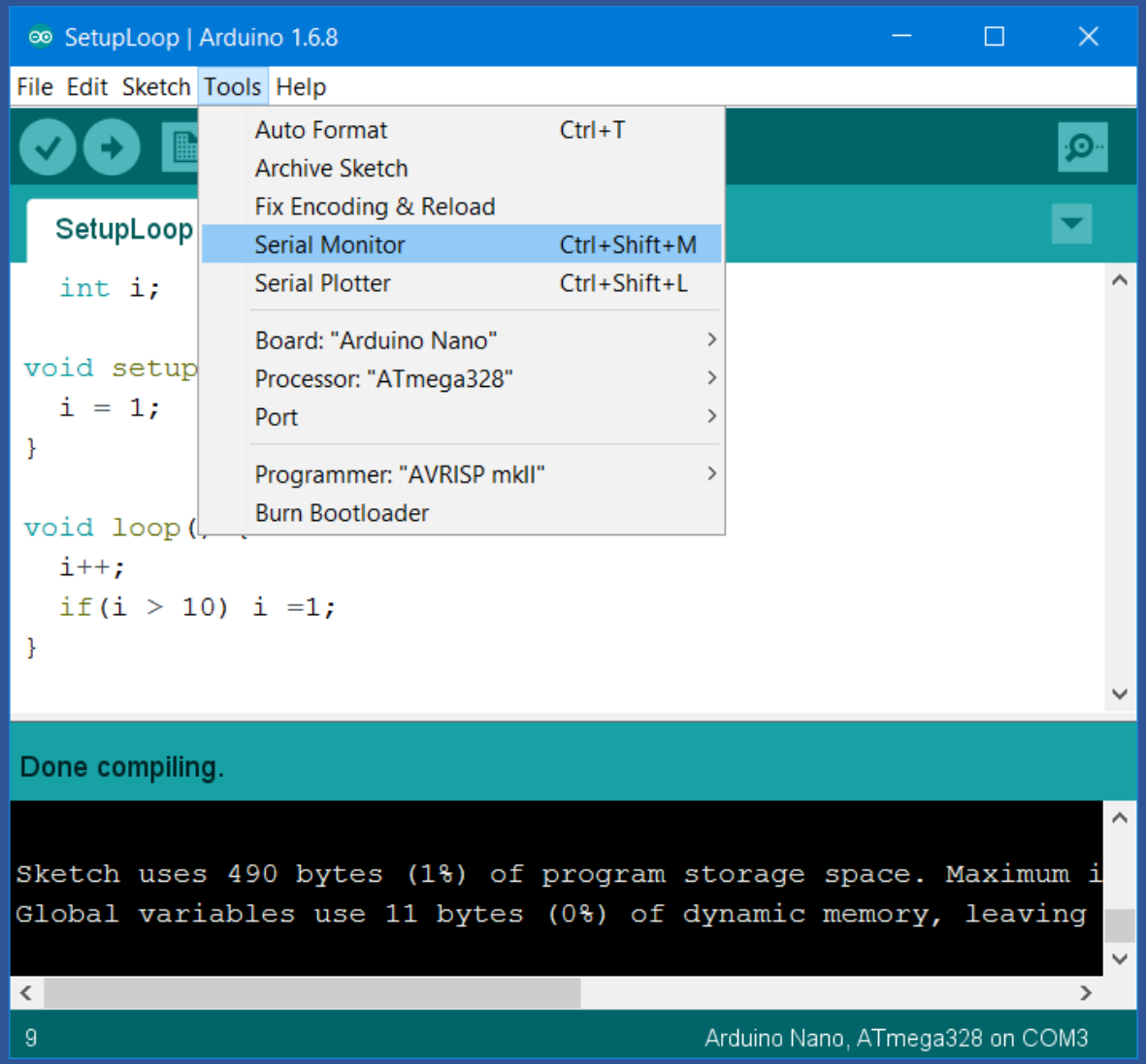
- Lab 1 Blinky: Write a program to blink an external LED
- Lab 2 BlinkyTrio: Blink 3 LEDs
- Lab3 Runner: Red LED on .5sec and off, Yellow LED on .5sec and off, Green LED on .5sec and off, repeat all over.

More on Sketch Language

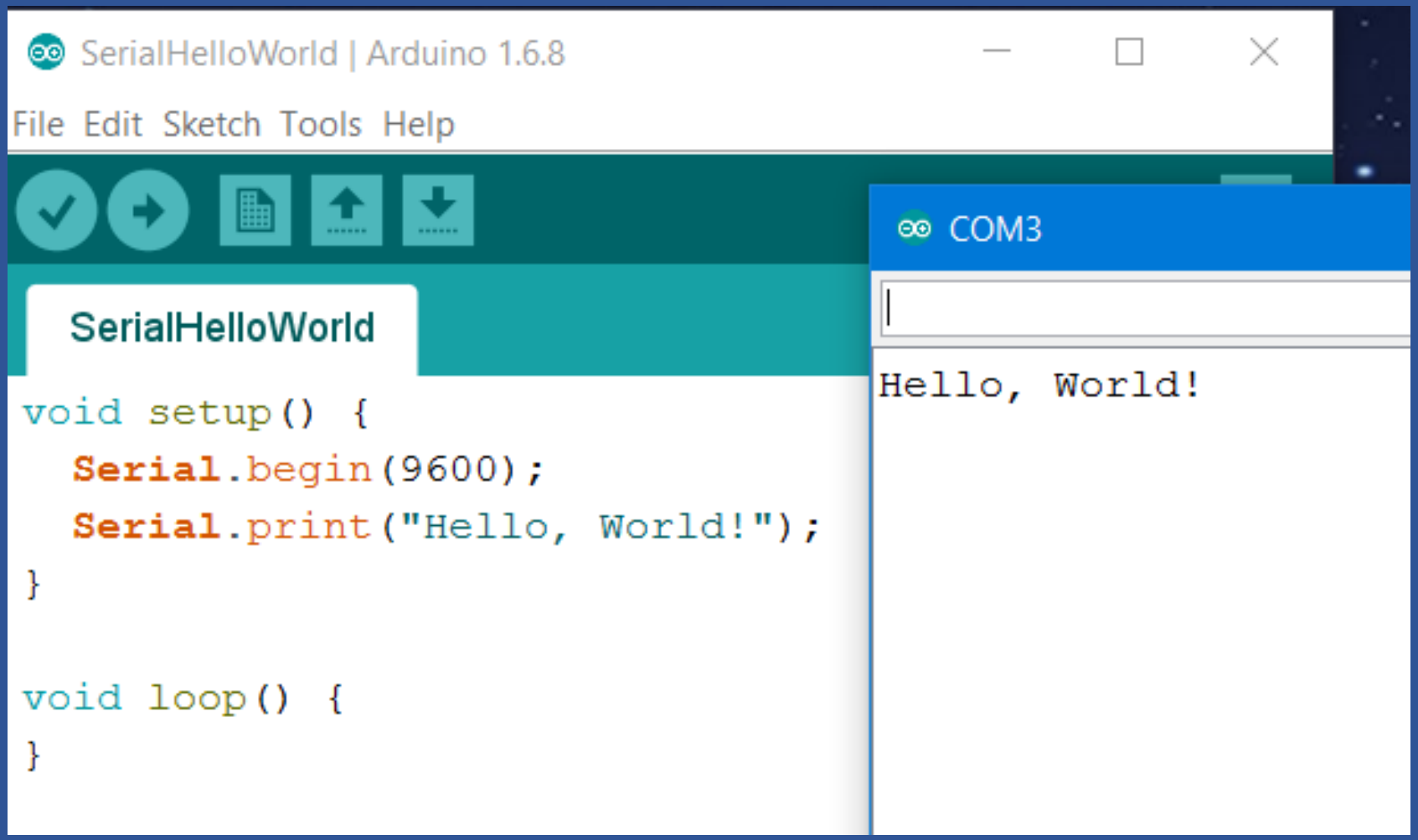
- Language Reference

https://www.arduino.cc/en/Reference/HomePage

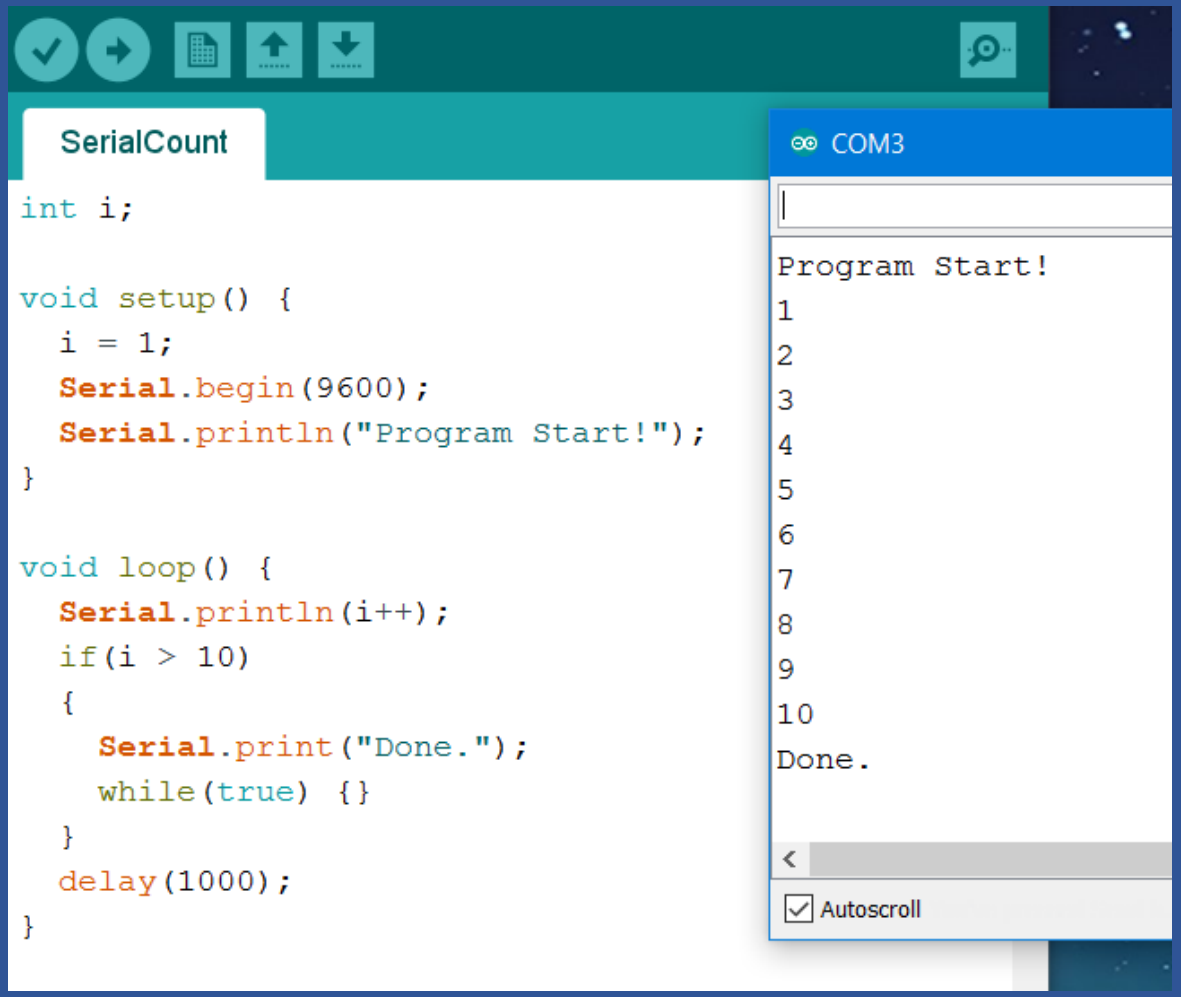
Programing and debugging



SerialHelloWorld



Serial Count



Serial.print()

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(78); //gives "78"
  Serial.println(1.23456); //gives "1.23"
  Serial.println('N'); //gives "N"
  Serial.println("Hello world."); //gives "Hello world."
  Serial.println(78, BIN); //gives "1001110"
  Serial.println(78, OCT); //gives "116"
  Serial.println(78, DEC); //gives "78"
  Serial.println(78, HEX); //gives "4E"
  Serial.println(1.23456, 0); //gives "1"
  Serial.println(1.23456, 2); //gives "1.23"
  Serial.println(1.23456, 4); //gives "1.2346"
  while(true){}
}
```

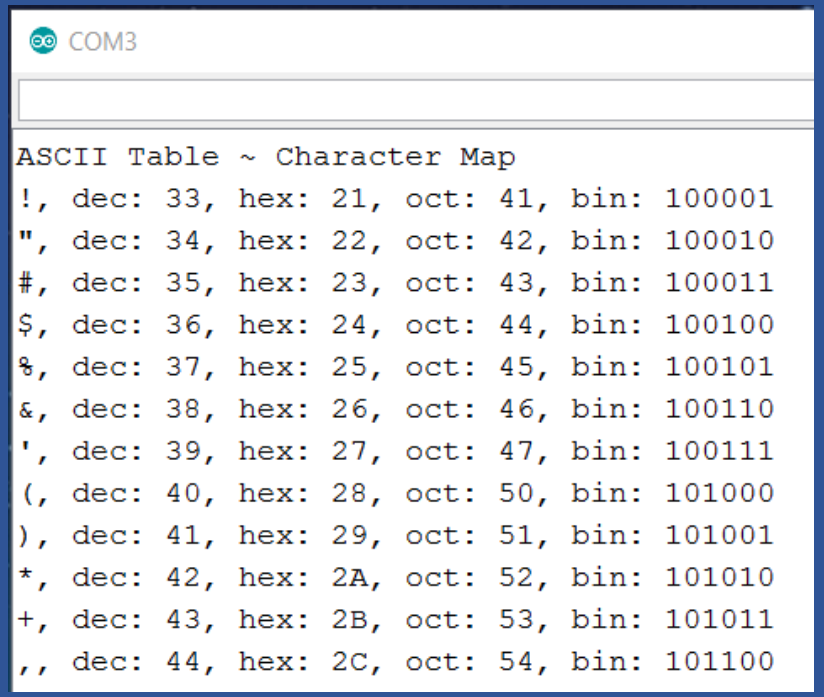
Serial Echo

SerialEcho §

```
void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    Serial.write(Serial.read());
  }
}
```

Serial Lab

- Lab1: AscTable: Print out ASCII Table
- Lab2: ShowChrAsc: Get a letter from Serial input and Send out one line of ASCII information



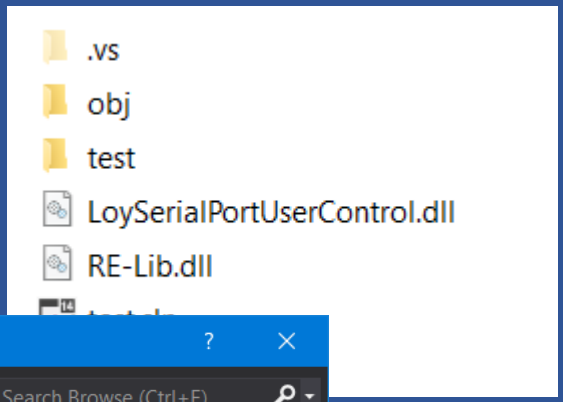
Arduino and C#

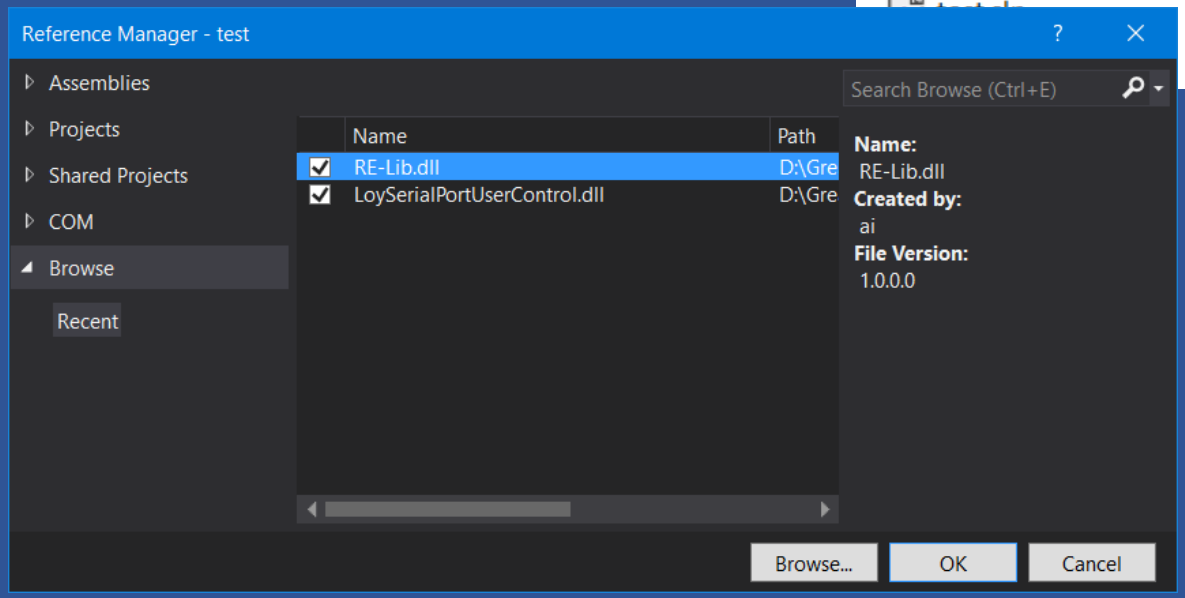
- Arduino send message to C# Win App
- Arduino read message from C# WinApp
- Arduino execute command from C# Win App

Develop C# Win App to Read Message from Arduino

1. Create New C# WinForm Project Copy and paste "LoySerialPortUserControl.dll" to project's folder

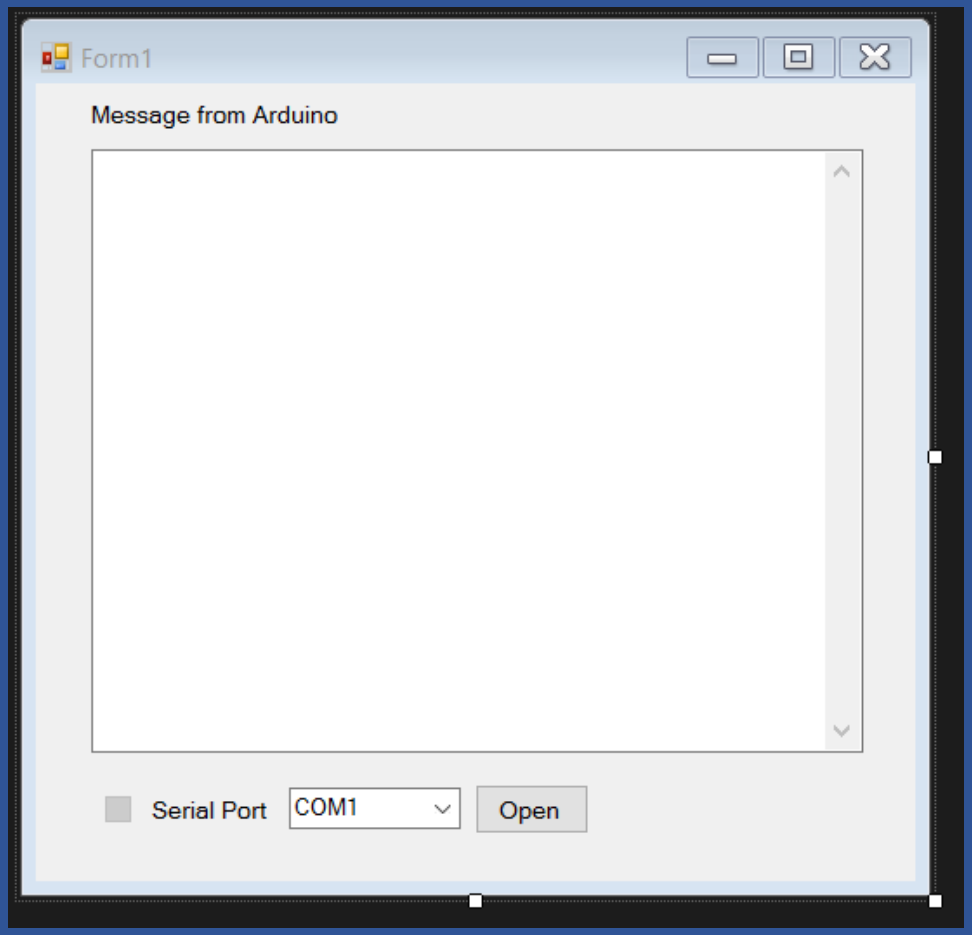
2. Add Reference to files





3. Add LoySerialPortUserControl.dll to Form1

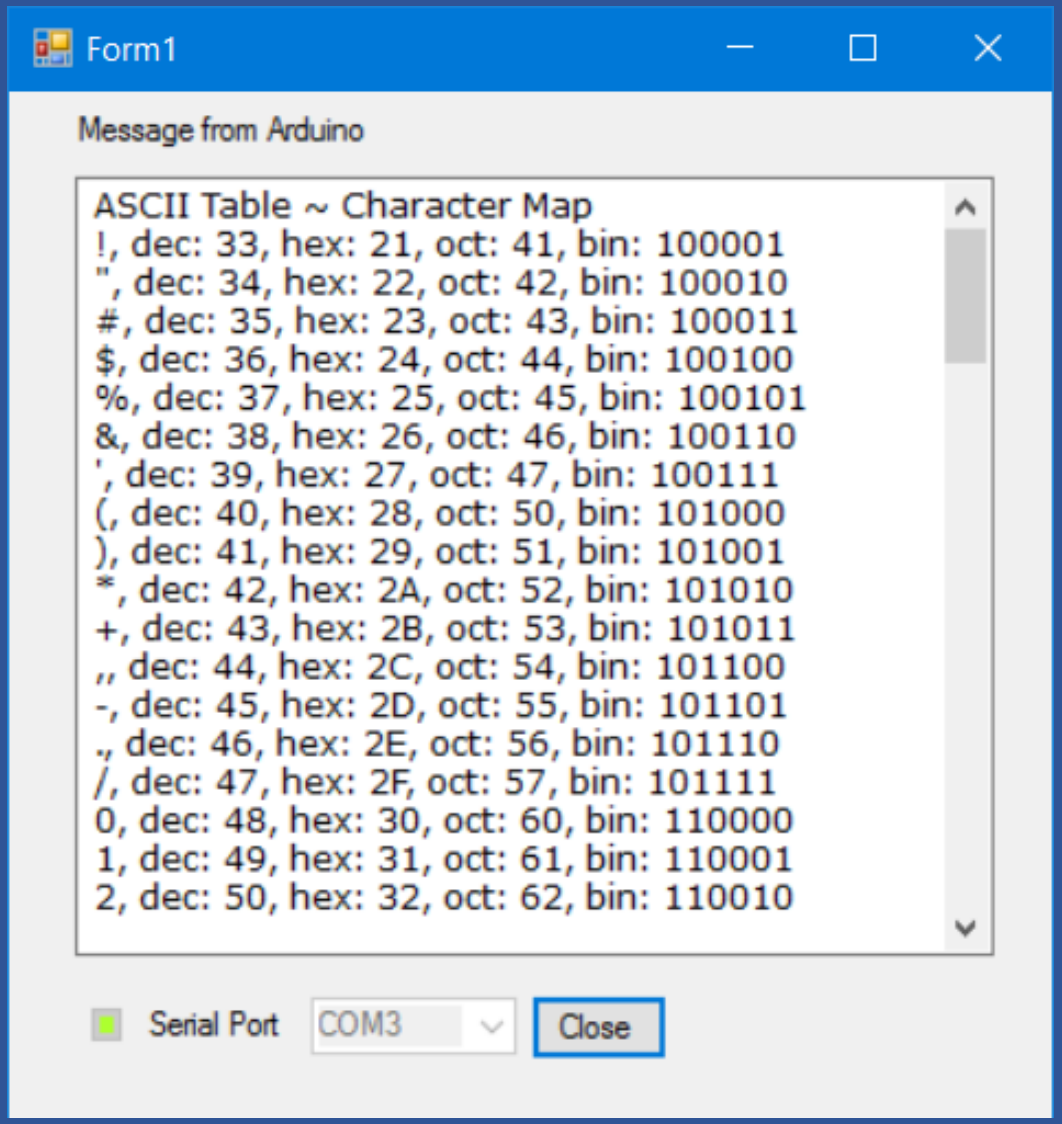
4. Add TextBox to Form1. Name = TextBoxRx



5.Add below code to Form1

```
using System;
using System.Windows.Forms;

namespace test
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
            loySerialPortUc1.OnDataReceived
                += LoySerialPortUc1_OnDataReceived;
        }

        1 reference
        private void LoySerialPortUc1_OnDataReceived(string recieveString)
        {
            textBoxRX.Invoke(new Action(() =>
            { textBoxRX.AppendText(recieveString); }));
        }
    }
}
```

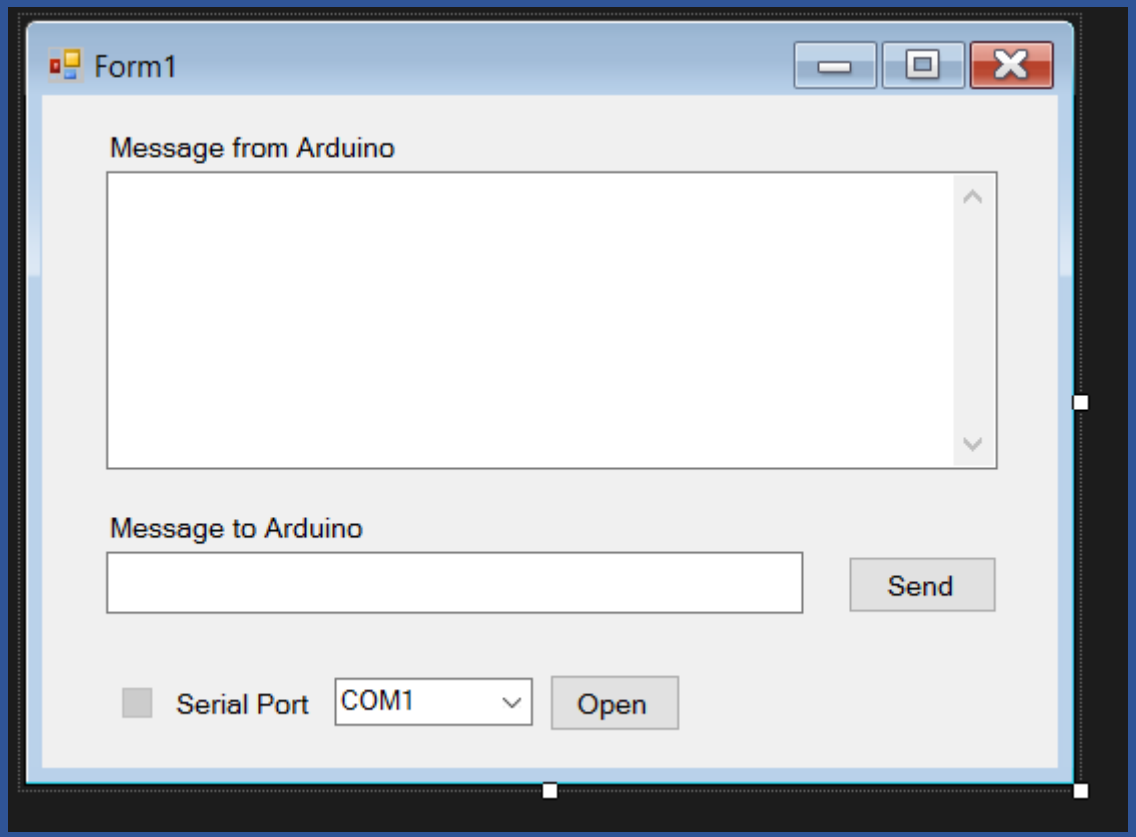
6.Test Program using Arduino AscTable code



Develop C# Win App to Write Message to Arduino

7. Add a TextBox to Form1. Name = "TextBoxTx"

8. Add a button to Form1. Name = buttonSend



9.Add below code to Form1

```
        1 reference
21      private void buttonSend_Click(object sender, EventArgs e)
22      {
23          loySerialPortUc1.Write(textBoxTx.Text);
24      }
25  }
26  }
```

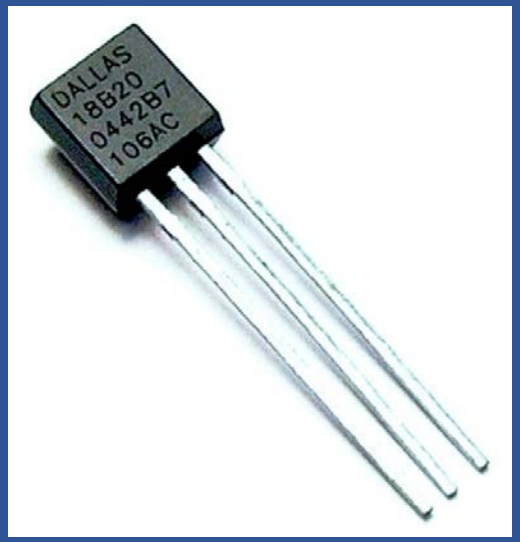
10. Test Program using Arduino SerialEcho.

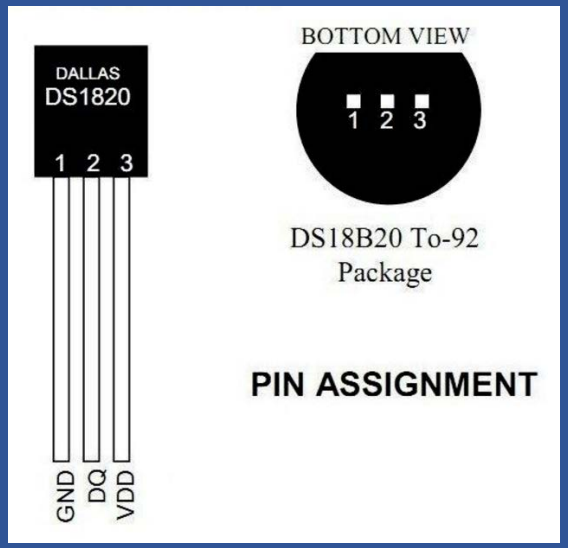
Arduino <--> PC communication Lab

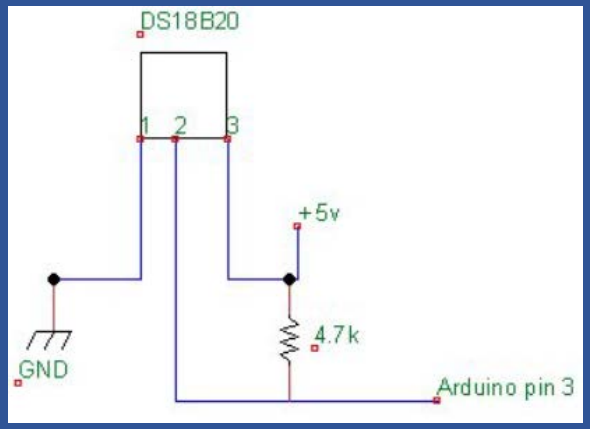
1. Wait for PC message
2. When receive message “turn on LED 1”, turn on LED at pin D2
3. When receive message “turn off LED 1”, turn off LED at pin D2
4. When receive message “turn on LED 2”, turn on LED at pin D3
5. When receive message “turn off LED 1”, turn off LED at pin D3

Getting data from Sensors

Temperature



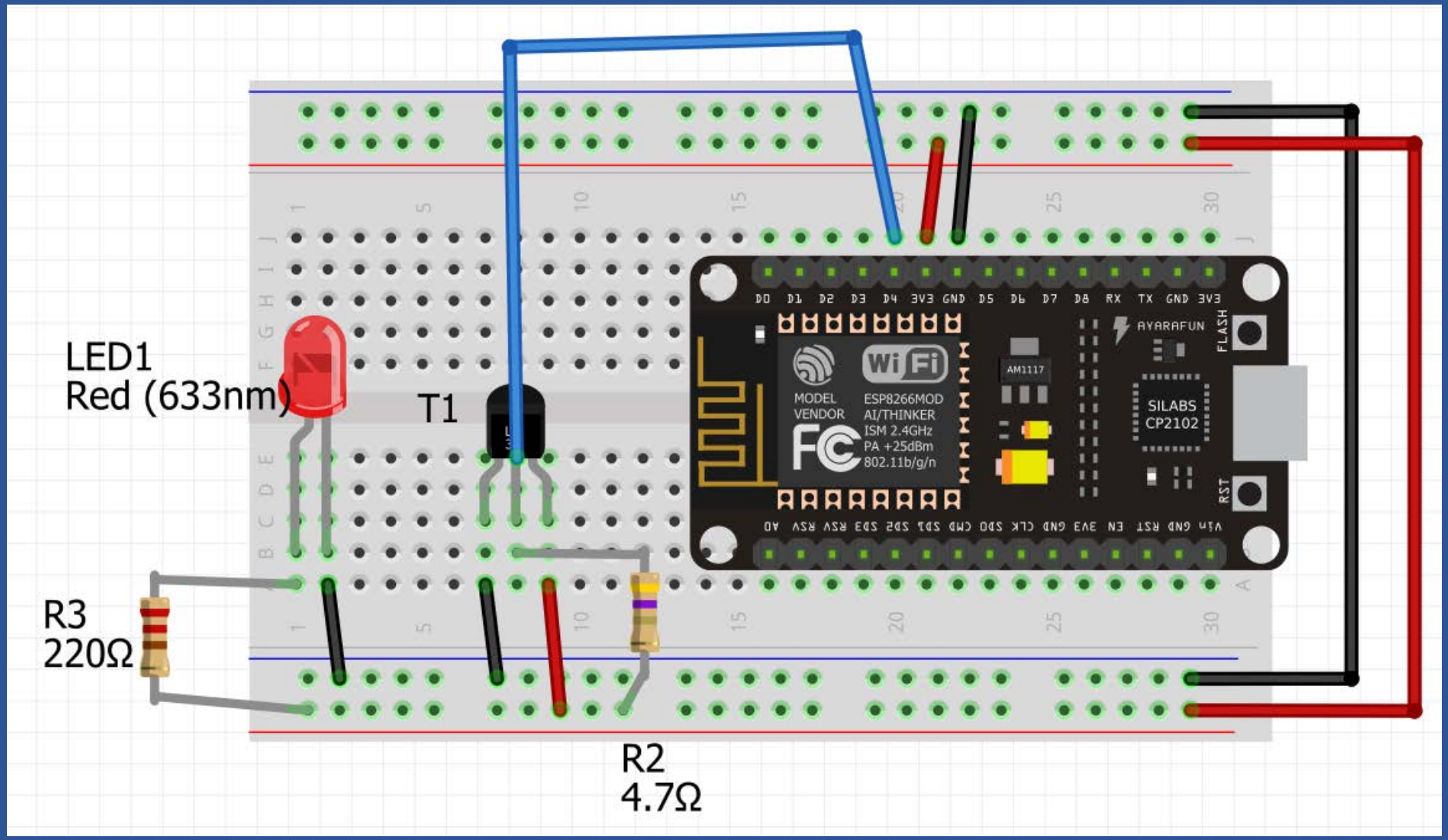




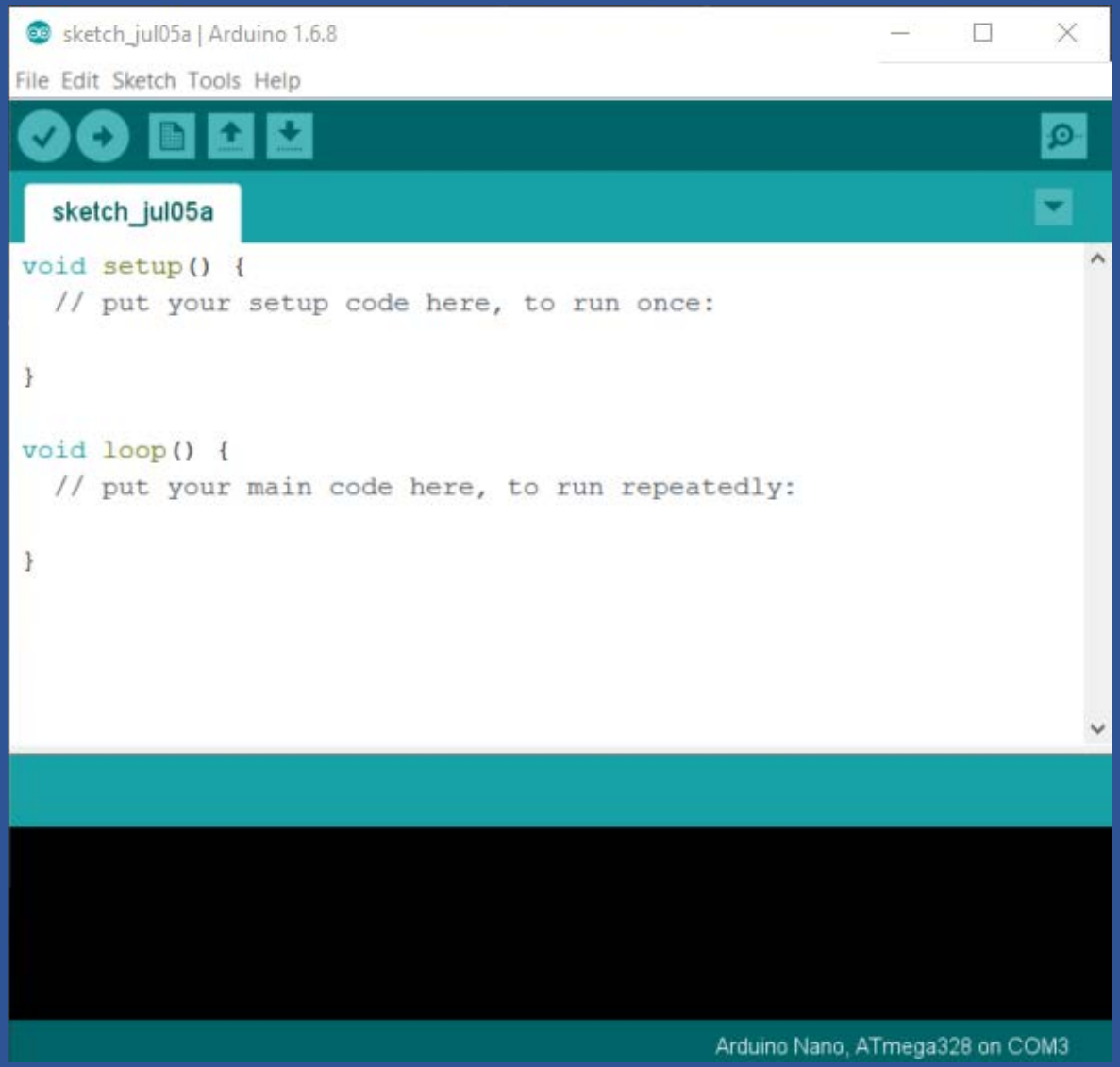
DS18B20 Programmable Resolution 1-Wire Digital Thermometer

Benefits and Features

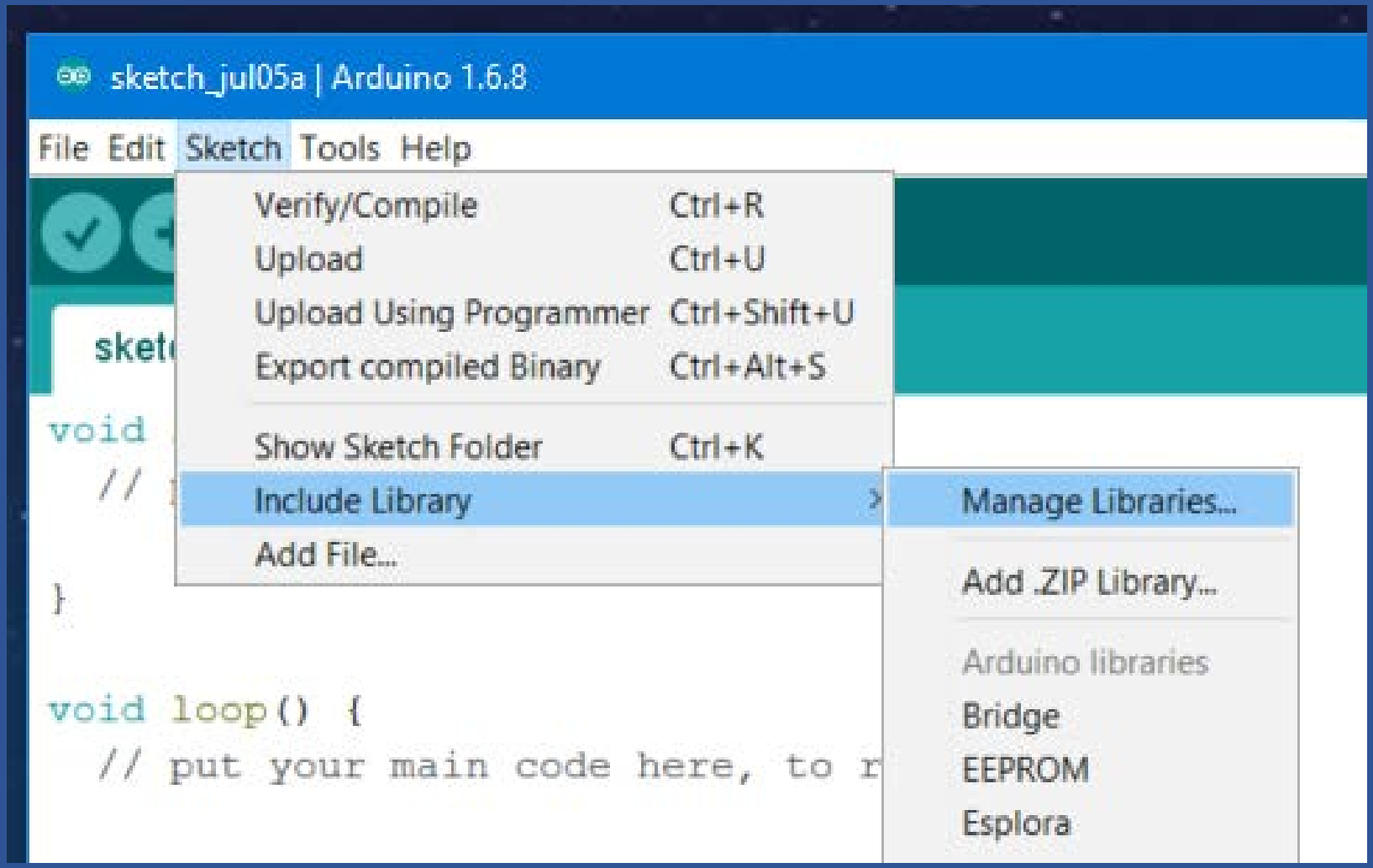
- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
- Measures Temperatures from-55°C to +125°C (-67°F to +257°F)
- Programmable Resolution from 9 Bits to 12 Bits
- No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)


LED1
Red (633nm)
T1
R3
220Ω
R2
4.7Ω
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
AYARAFUN
FLASH
RST
AM1117
MODEL
VENDOR
ESP8266MOD
AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
SILABS
CP2102

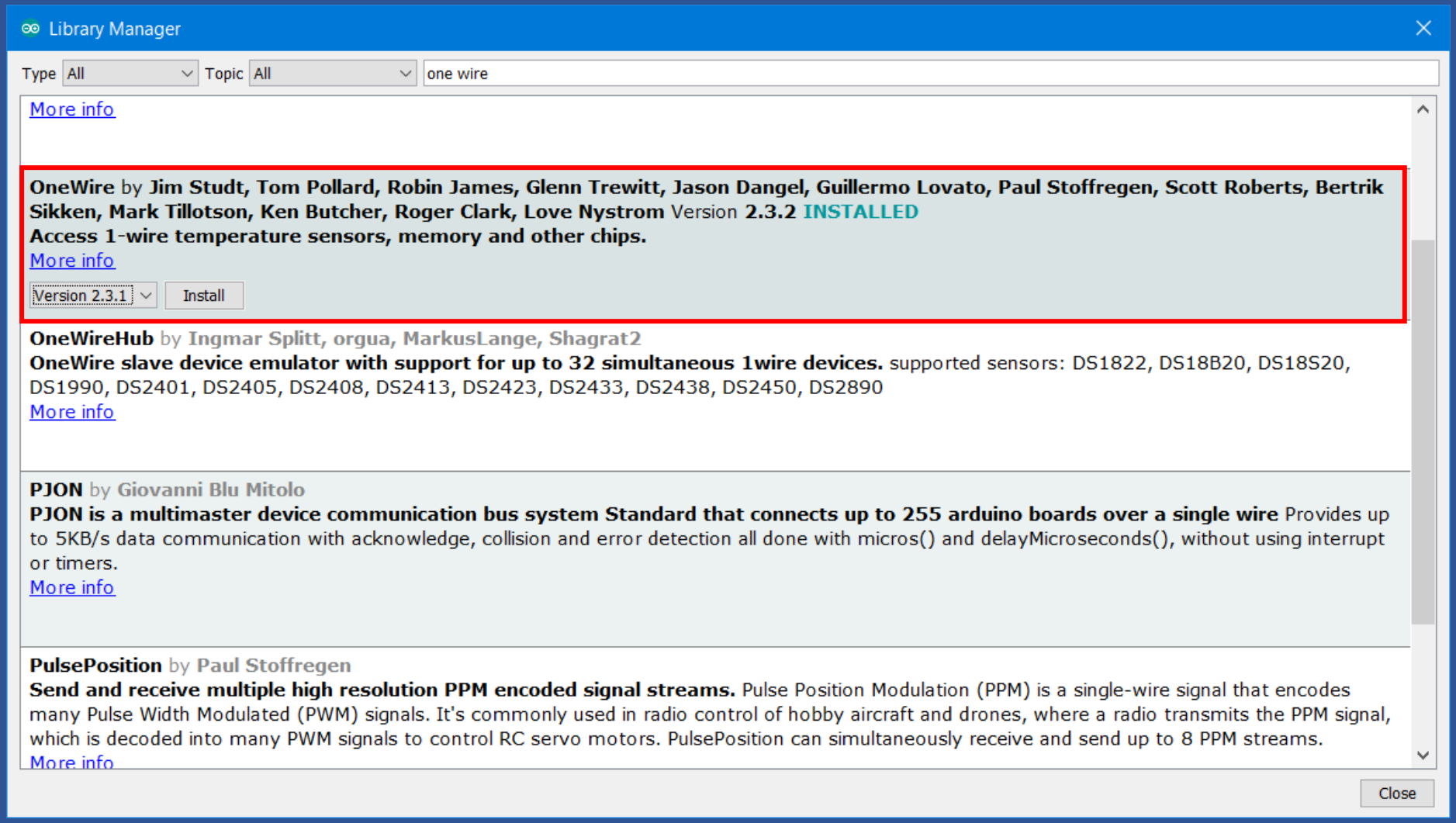
Create New Arduino Project



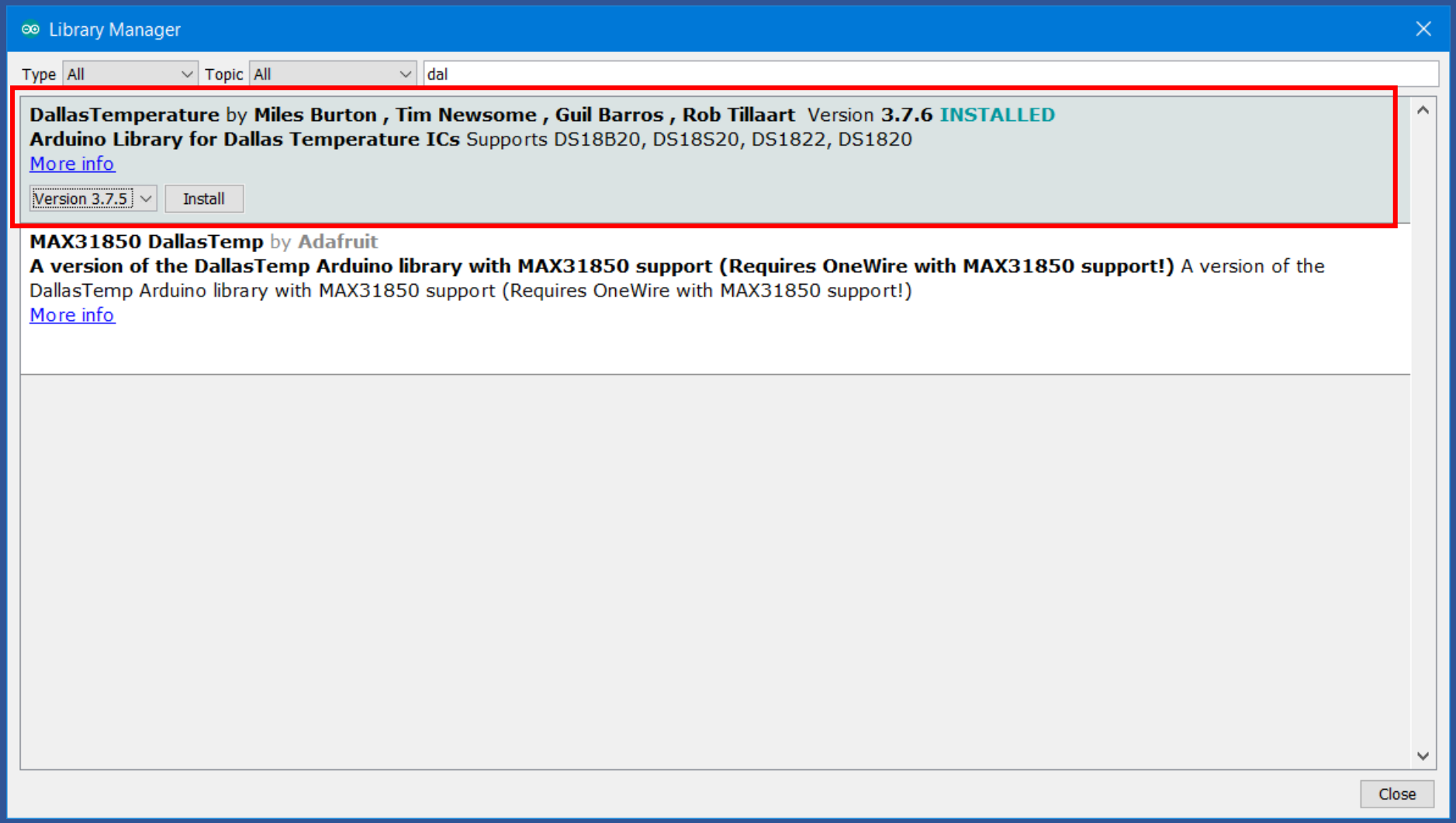
Search / Include Library / Manage Libraries...



Search “one wire” / Select Version / Install



Search “DallasTemperature” / Select Version / Install

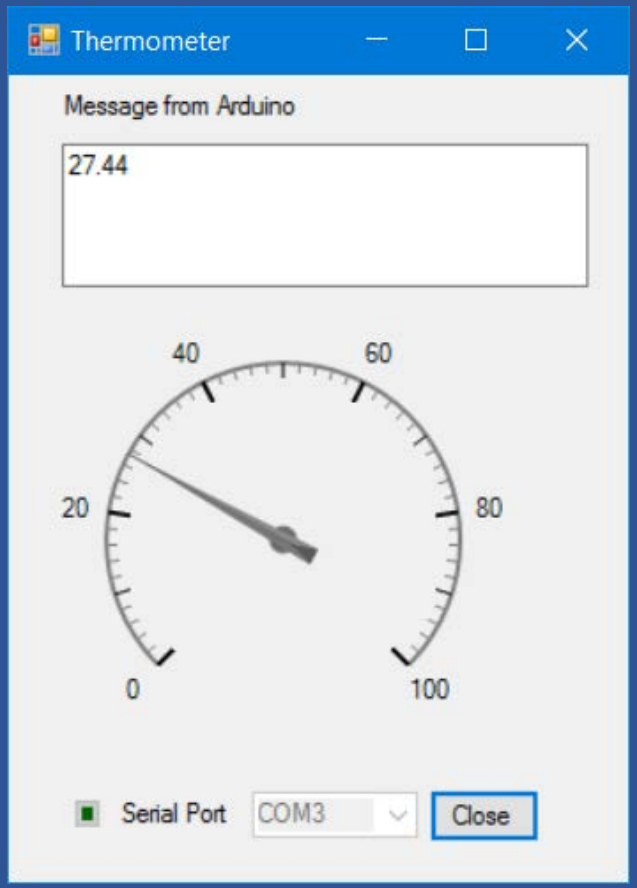


Temperature

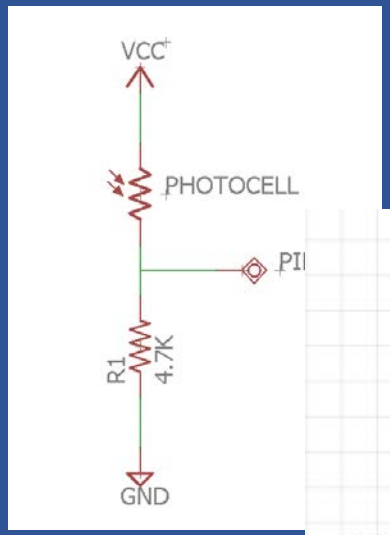
```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_PIN 5  // connect ds18B20 to pin D5
OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  sensors.requestTemperatures(); // Send the command to get temperatures
  Serial.print(sensors.getTempCByIndex(0));
  delay(5000);
}
```

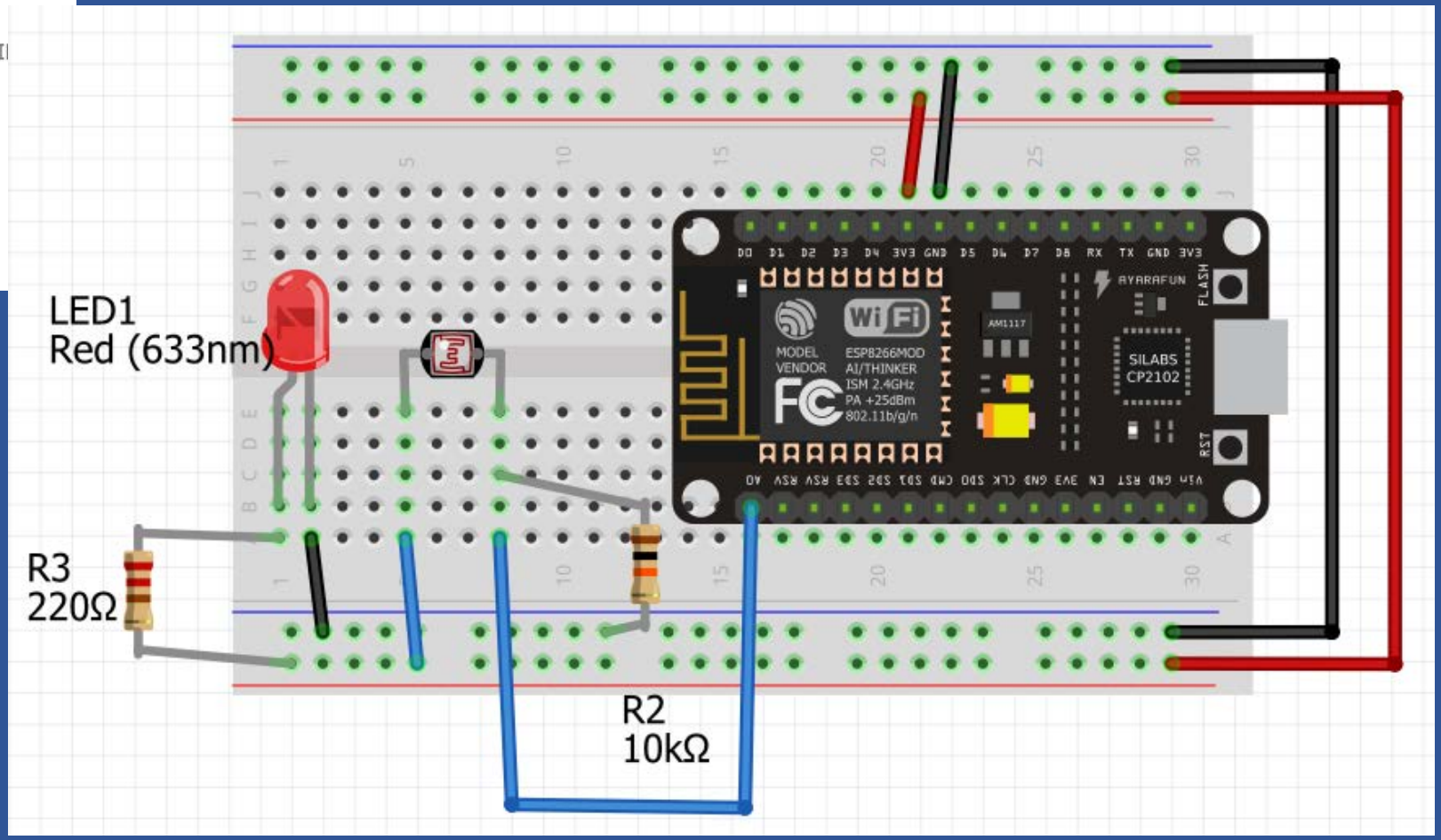
Temperature Lab

1. Lab1: Thermometer, show temperature in WinForm with a gauge.
2. Lab2: TemClound, send temperature to Cloud.



Light Sensor



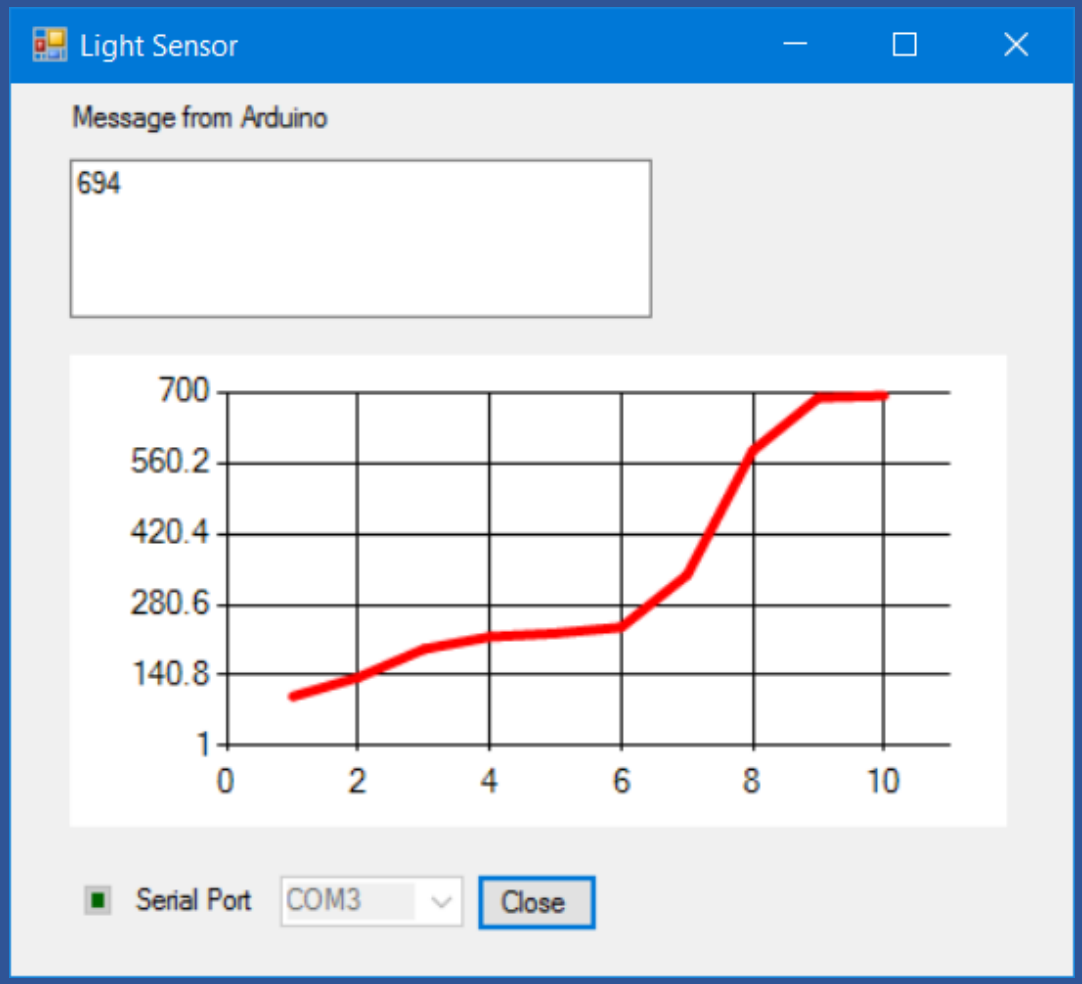


Arduino Light1

Light1

```
const int analogInPin = A0;
int sensorValue = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(analogInPin);
  Serial.print(sensorValue);
  delay(100);
}
```

Light sensor WinForm



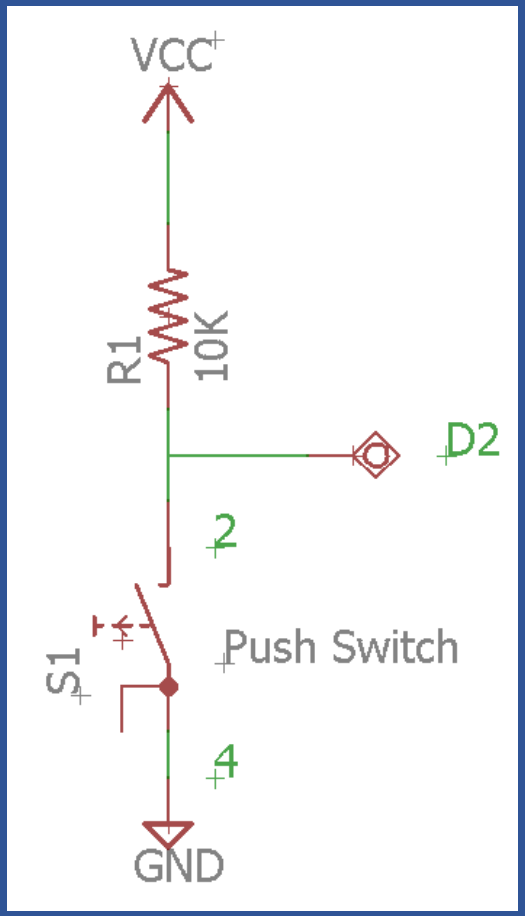
Light sensor Lab

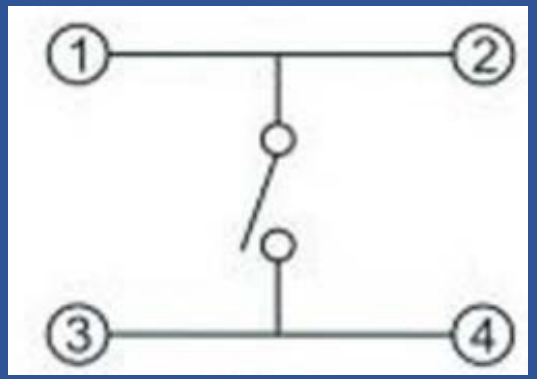
1. Modify Light sensor WinForm to rise audio alarm if the value the beyond the specified value

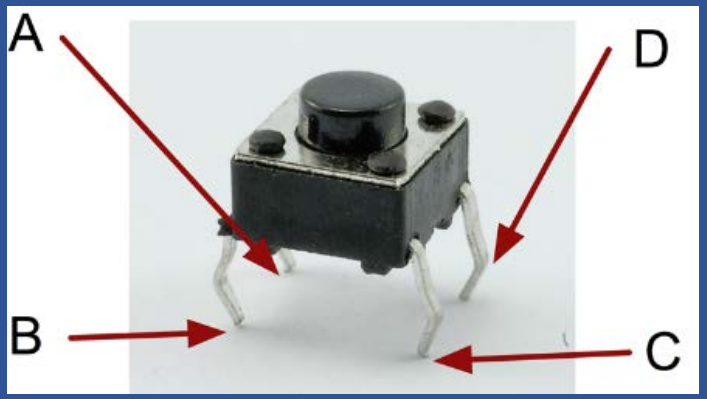
2. Modify Light sensor WinForm to show other type of chart

3. Modify Light sensor WinForm to save data to log file when the value the beyond the specified value

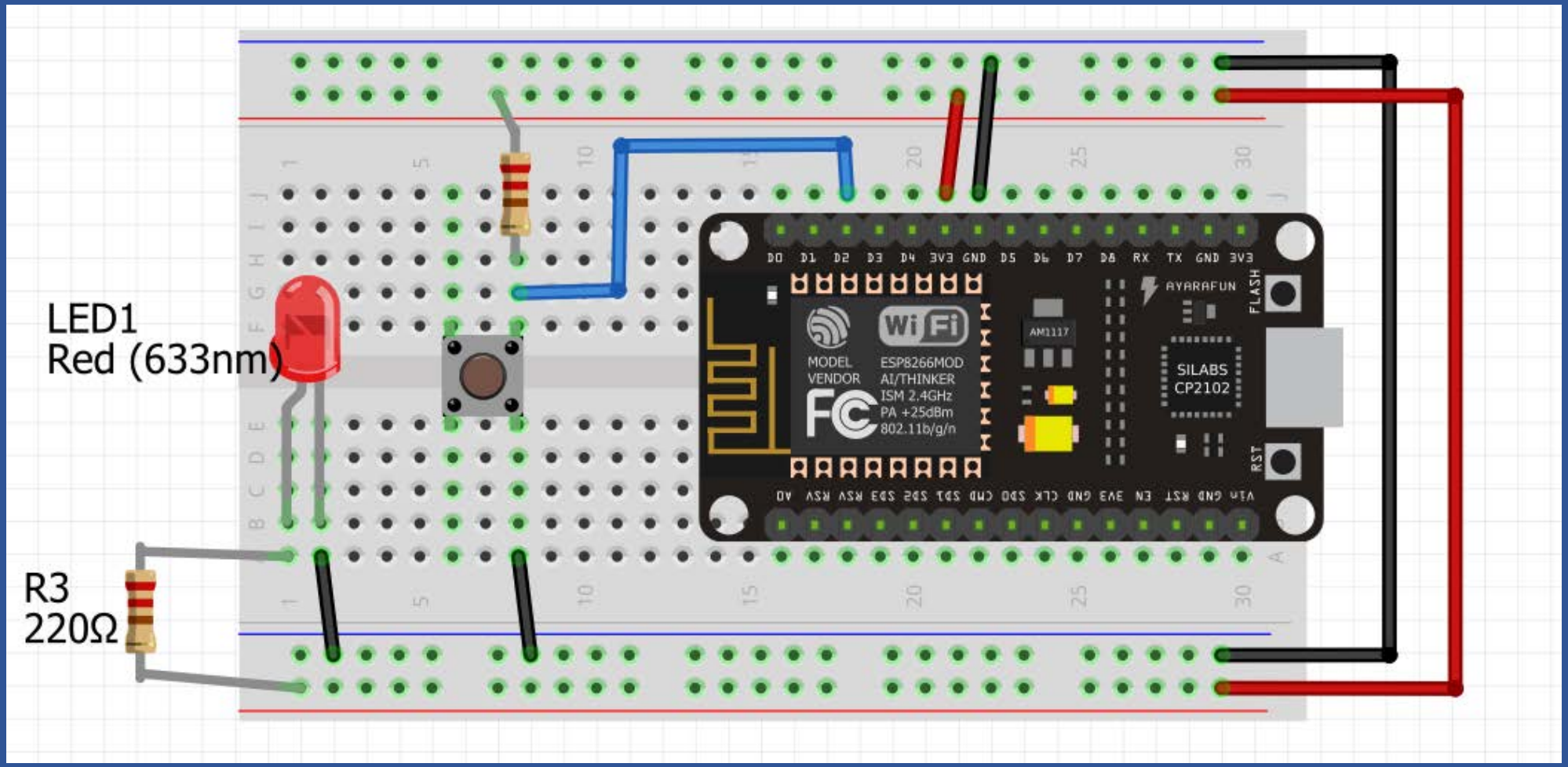
Physical sensors





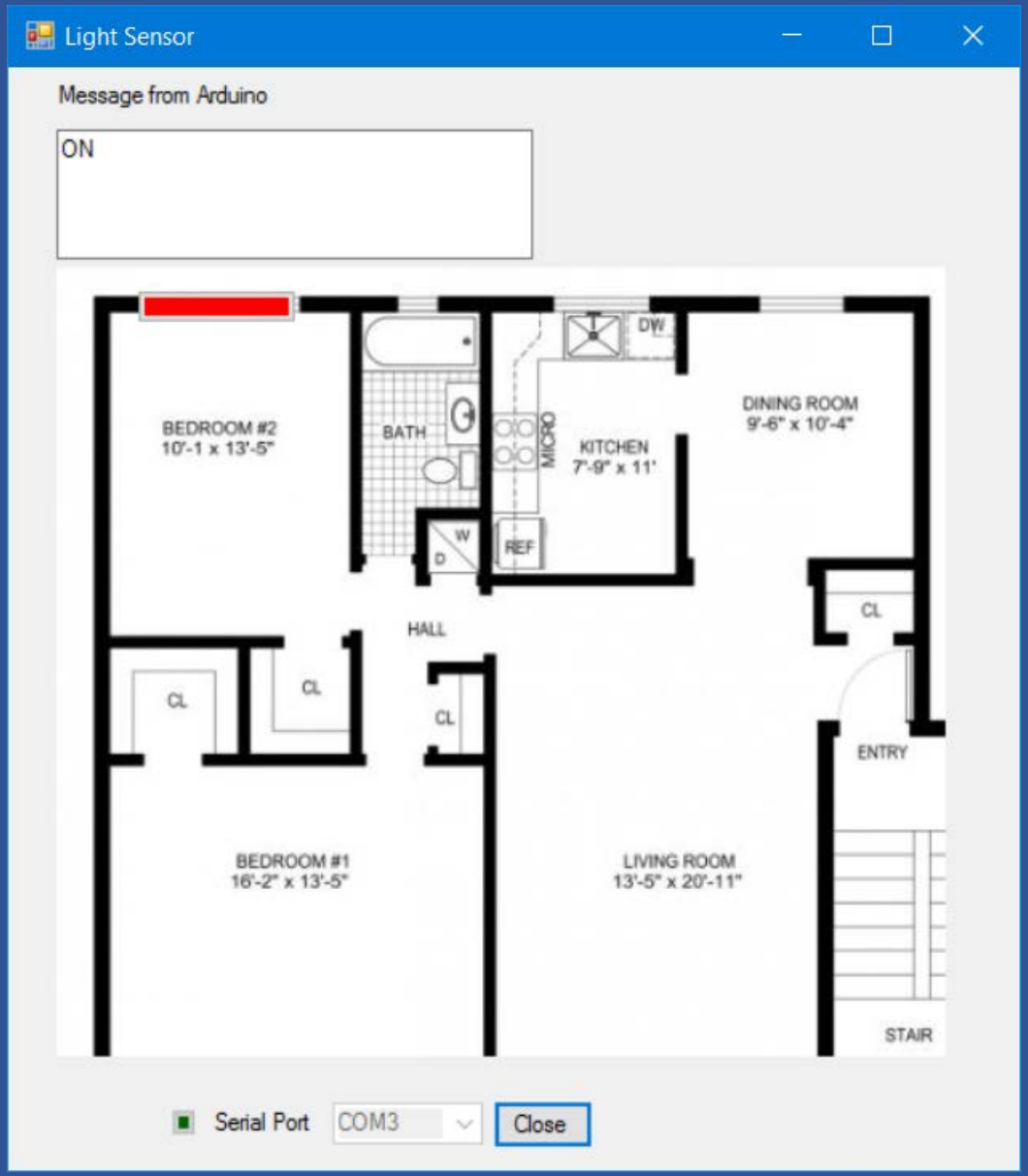


Mini Pushbutton Switch

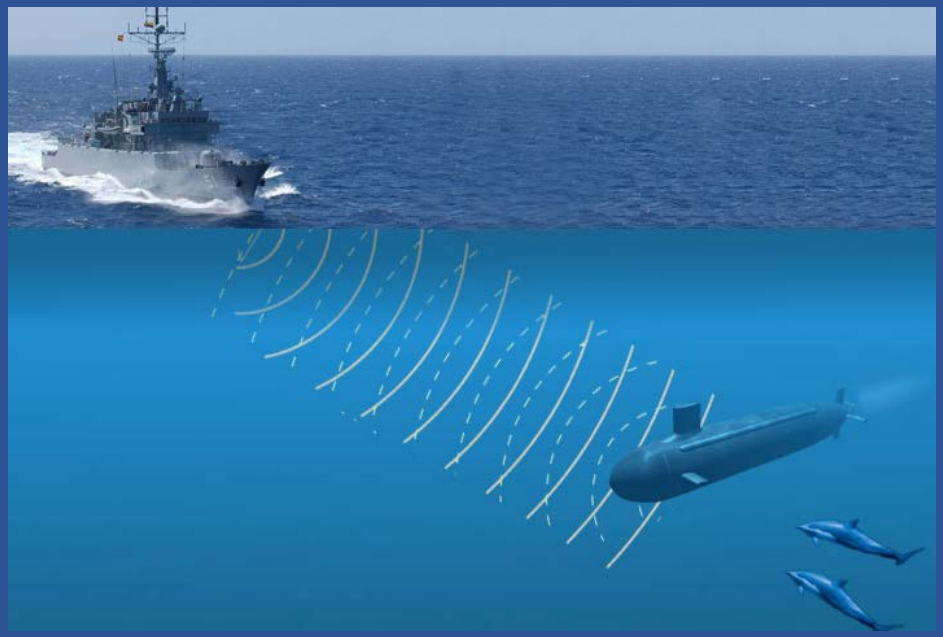


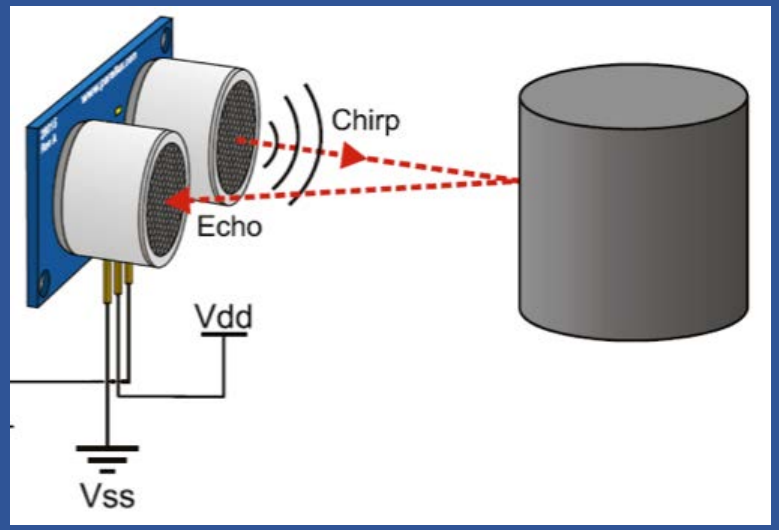
Switch

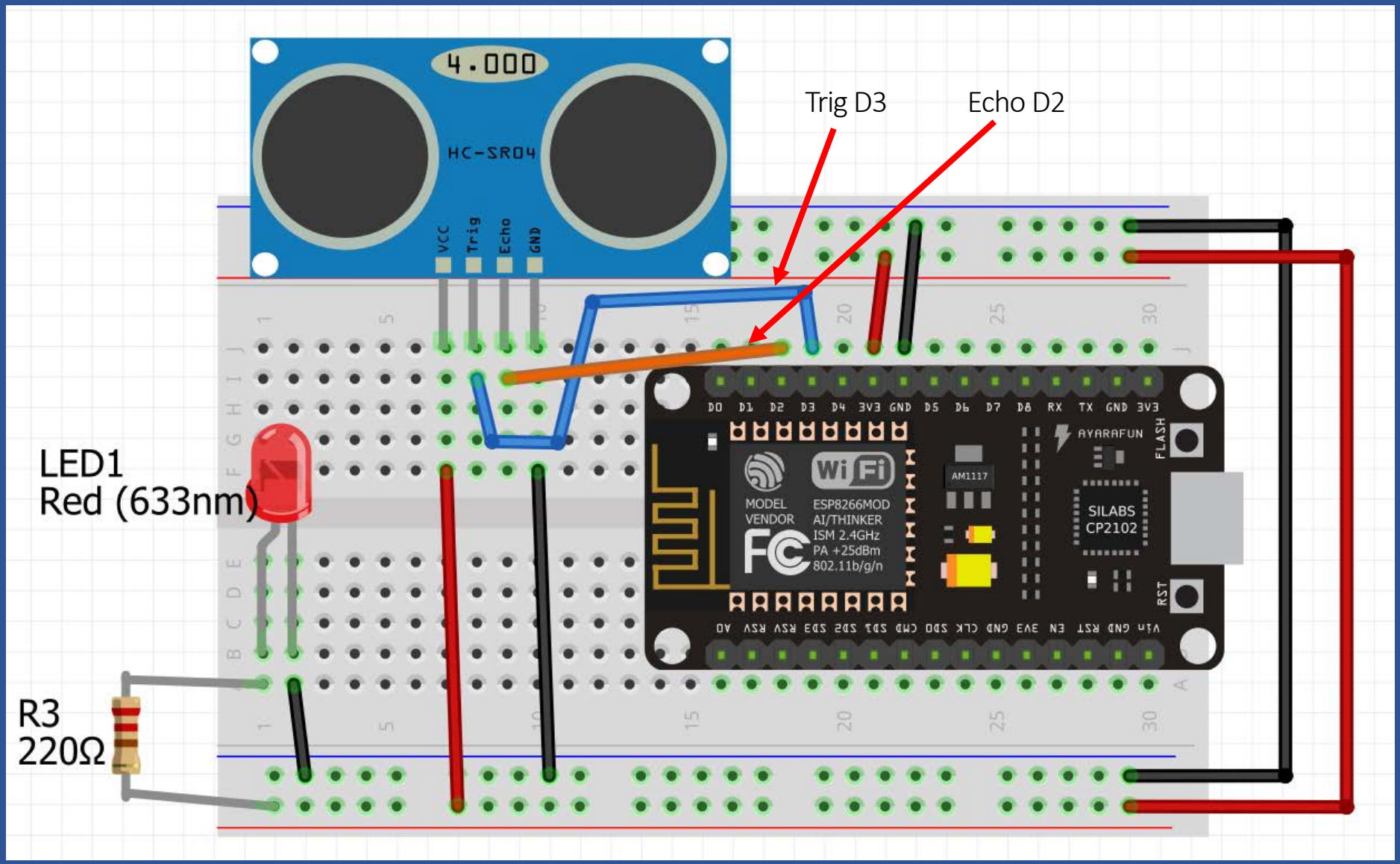
```
const int buttonPin = 2;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  if(digitalRead(buttonPin) == HIGH)
    Serial.print("OFF");
  else
    Serial.print("ON");
  delay(100);
}
```


Light Sensor
Message from Arduino
ON
BEDROOM #2
10'-1 x 13'-5"
BATH
MICRO
KITCHEN
7'-9" x 11'
DW
DINING ROOM
9'-6" x 10'-4"
W
D
REF
CL
HALL
CL
CL
CL
ENTRY
BEDROOM #1
16'-2" x 13'-5"
LIVING ROOM
13'-5" x 20'-11"
STAIR
Serial Port
COM3
Close

Ultrasonic



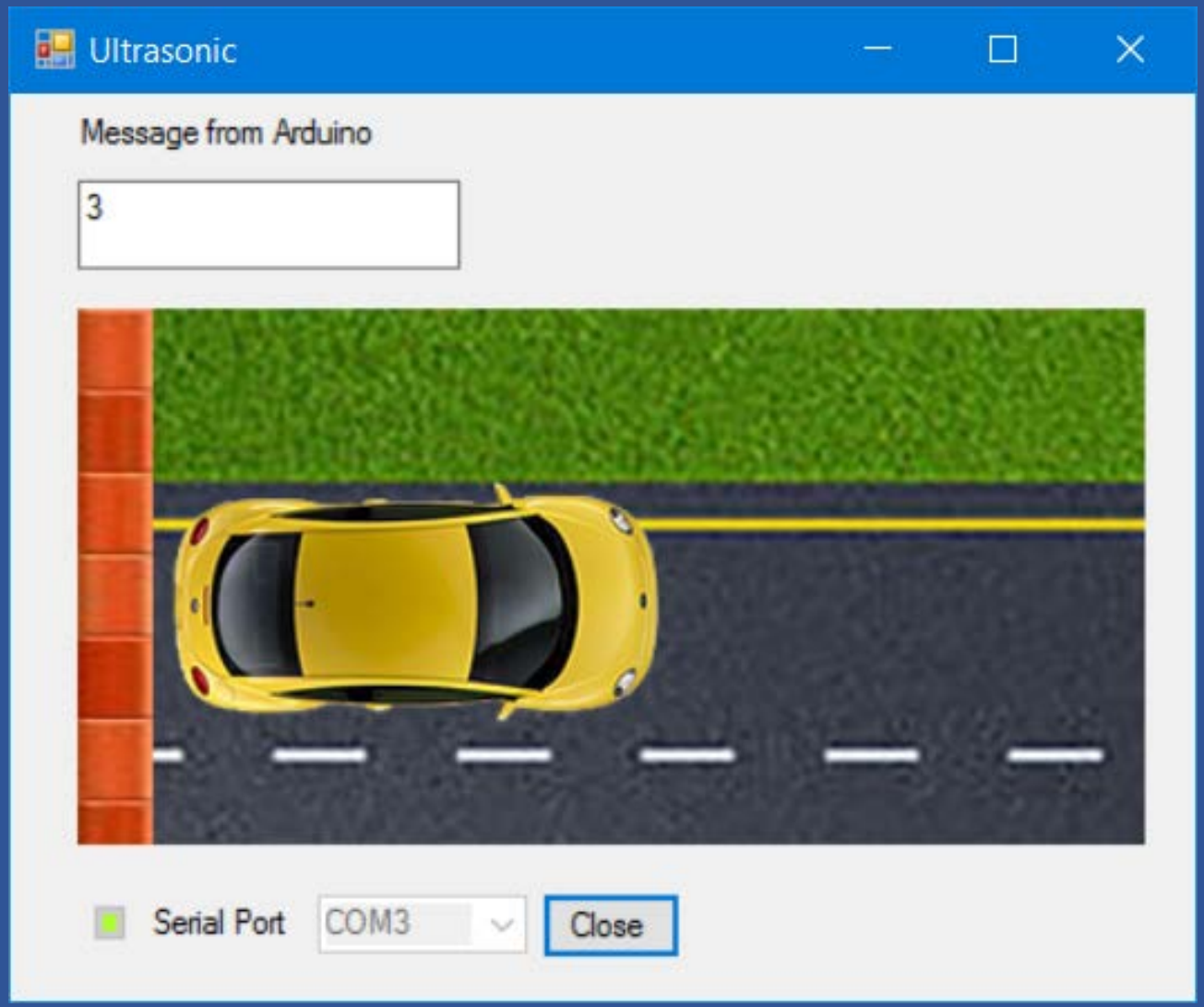



4.000
HC-SR04
VCC
Trig
Echo
GND
Trig D3
Echo D2
LED1
Red (633nm)
R3
220Ω
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
AYARAFUN
FLASH
Wi Fi
MODEL
VENDOR
ESP8266MOD
AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
AM1117
SILABS
CP2102
RST

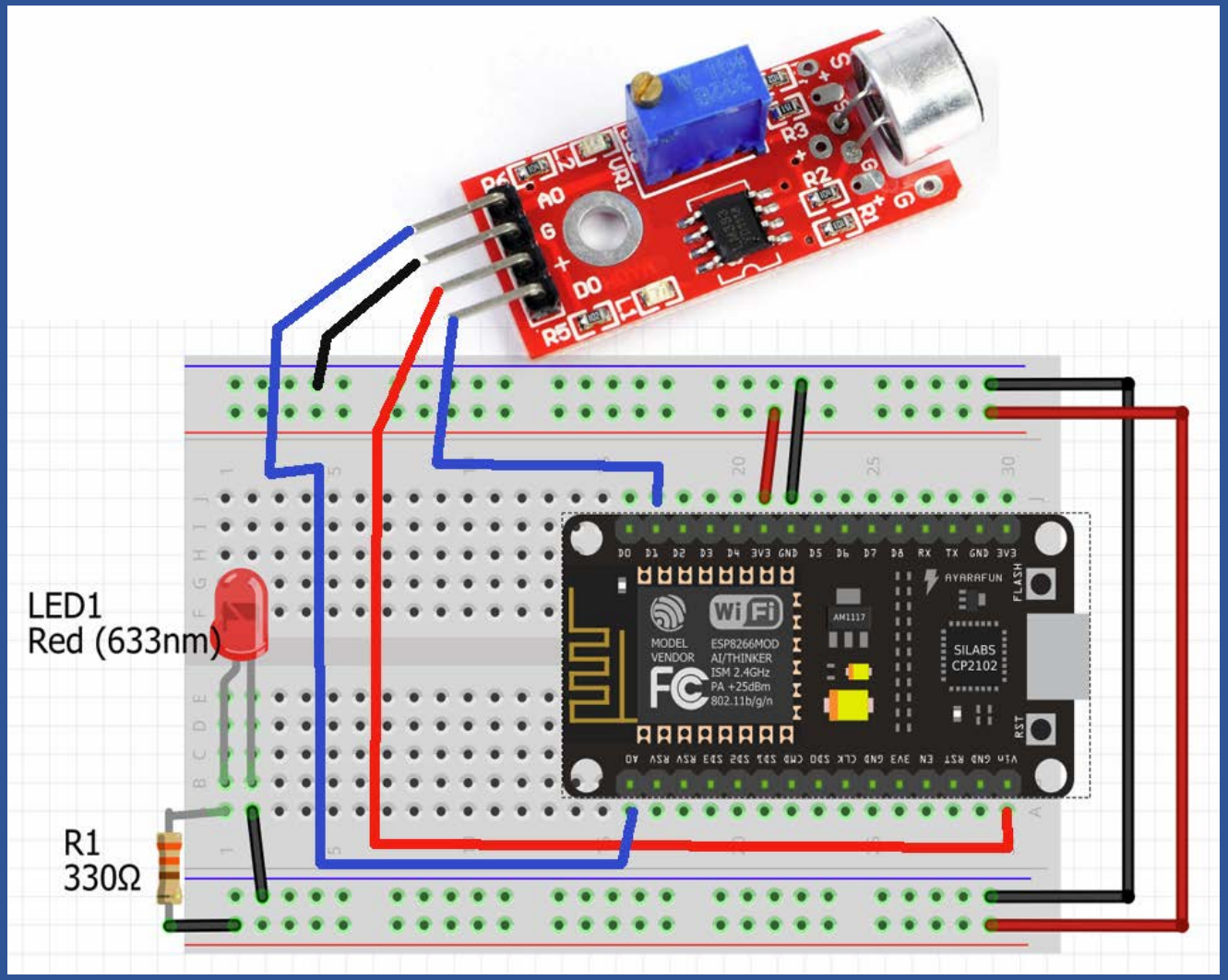
Arduino Ultrasonic Code

```
#define TRIGGER_PIN  D1
#define ECHO_PIN     D2

void setup() {
  Serial.begin (9600);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUILTIN_LED, OUTPUT);
}

void loop() {
  long duration, distance;
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration/2) / 29.1;
  Serial.print(distance);
  Serial.println(" cm");

  digitalWrite(BUILTIN_LED, HIGH);
  delay(100);
  digitalWrite(BUILTIN_LED, LOW);
  delay(100);
}
```

Ultrasonic demo WinForm App



Sound


LED1
Red (633nm)
R1
330Ω
AO
G
+
DO
R3
R2
R5
MODEL
VENDOR
ESP8266MOD
AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
SILABS
CP2102
AYARFUN
FLASH
RST

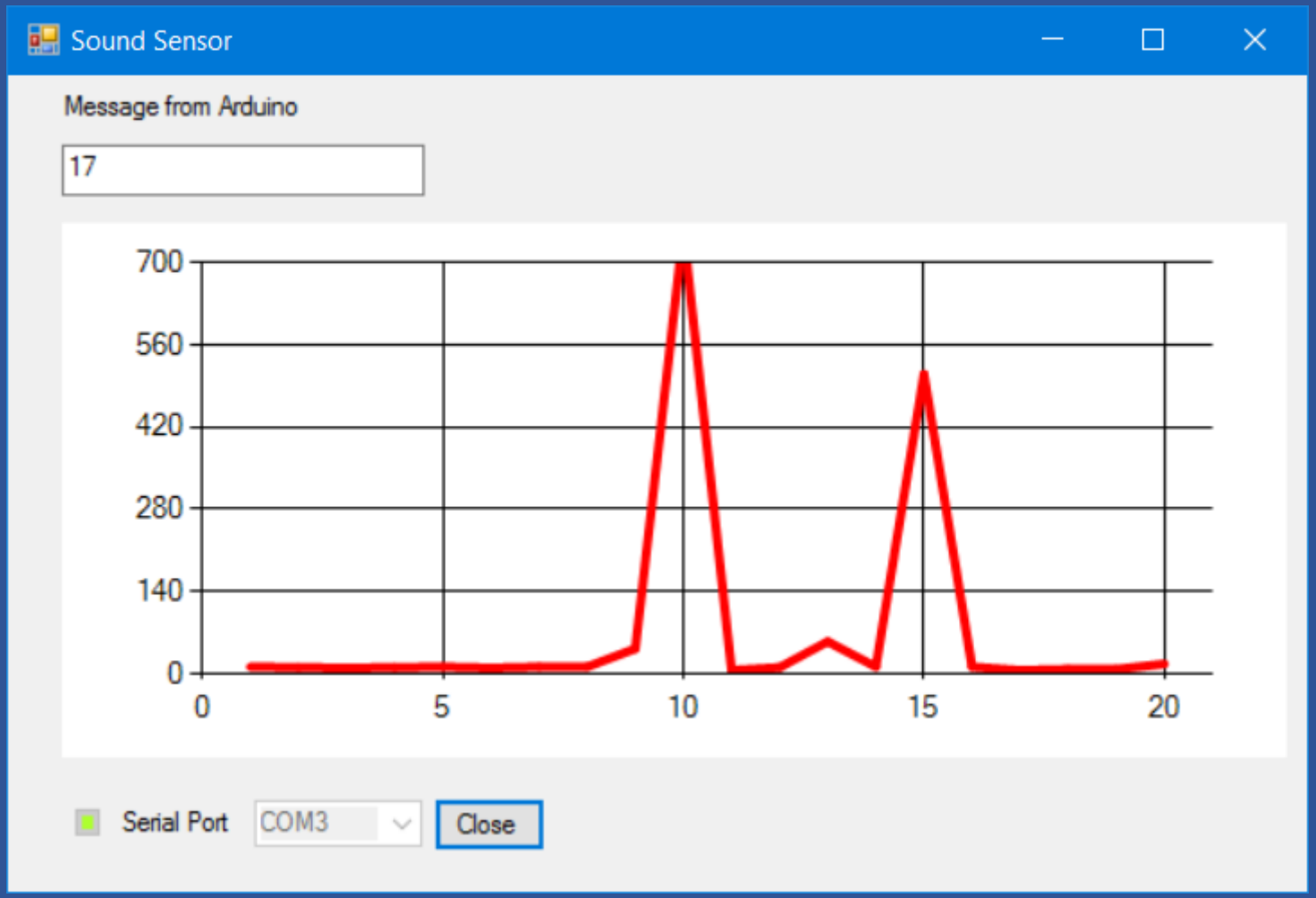
Arduino Sound Digital mode

```
int Led=BUILTIN_LED;

int mic =D1;
int val;
void setup()
{
    pinMode(Led,OUTPUT);
    pinMode(mic,INPUT);
}
void loop()
{
    val=digitalRead(mic);
    if(val==HIGH)
    {
        digitalWrite(Led,HIGH);
    }
    else
    {
        digitalWrite(Led,LOW);
    }
}
```

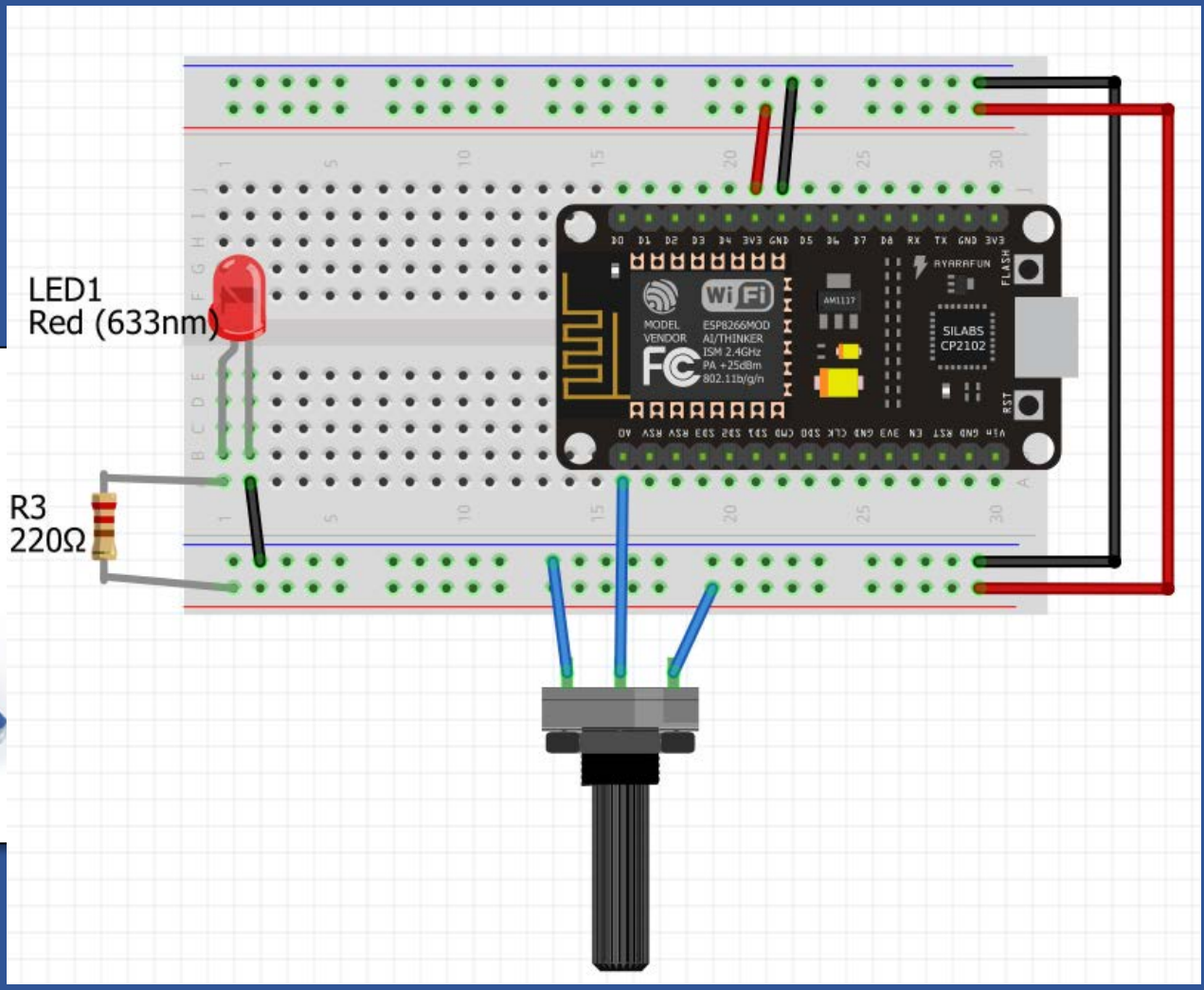
Arduino Sound Analog mode

```
int sensorPin = A0;
int ledPin =BUILTIN_LED;
int sensorValue = 4;

void setup ()
{
  pinMode (ledPin, OUTPUT);
  Serial.begin (9600);
}

void loop ()
{
  sensorValue = analogRead (sensorPin);
  Serial.println (sensorValue, DEC);
  digitalWrite (ledPin, HIGH);
  delay (sensorValue);
  digitalWrite (ledPin, LOW);
  delay (sensorValue);
}
```

WinForm Sound Sensor



Potentiometer



Potentiometer

```
/*
Show value of POT at serial terminal
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Potentiometer

```
1  //Blink on-board LED use external POT to set time interval blink time
2  // laploy july 2017
3
4  int sensorPin = A0;    // select the input pin for the potentiometer
5  int ledPin = 13;      // select the pin for the LED
6  int sensorValue = 0;  // variable to store the value coming from the sensor
7
8  void setup() {
9    // declare the ledPin as an OUTPUT:
10   pinMode(ledPin, OUTPUT);
11 }
12
13 void loop() {
14   // read the value from the sensor:
15   sensorValue = analogRead(sensorPin);
16   // turn the ledPin on
17   digitalWrite(ledPin, HIGH);
18   // stop the program for <sensorValue> milliseconds:
19   delay(sensorValue);
20   // turn the ledPin off:
21   digitalWrite(ledPin, LOW);
22   // stop the program for for <sensorValue> milliseconds:
23   delay(sensorValue);
24 }
```

Potentiometer

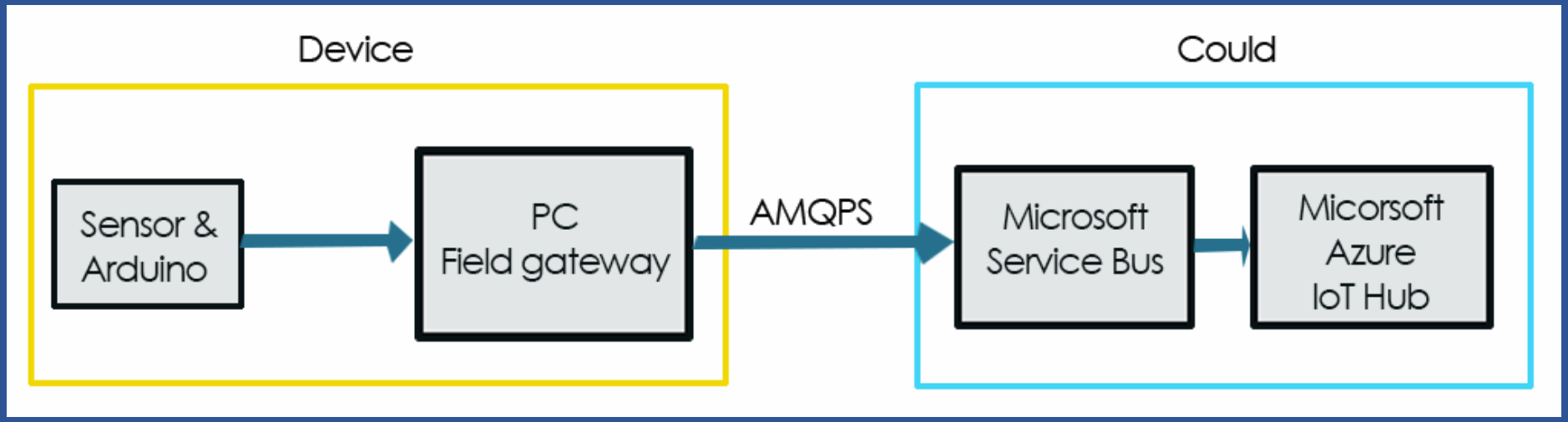
```
#define ROTARY_ANGLE_SENSOR A0
#define LED 13
#define ADC_REF 3.3
#define GROVE_VCC 3.3
#define FULL_ANGLE 300
void setup()
{
    Serial.begin(9600);
    pinsInit();
}
void loop()
{
    int degrees;
    degrees = getDegree();
    Serial.println("The angle between the mark and the starting position:");
    Serial.println(degrees);
    int brightness;
    brightness = map(degrees, 0, FULL_ANGLE, 0, 255);
    controlBrightness(brightness);
    delay(500);
}
void pinsInit()
{
    pinMode(ROTARY_ANGLE_SENSOR, INPUT);
    pinMode(LED,OUTPUT);
}
void controlBrightness(int brightness)
{
    analogWrite(LED,brightness);
}
int getDegree()
{
    int sensor_value = analogRead(ROTARY_ANGLE_SENSOR);
    float voltage;
    voltage = (float)sensor_value*ADC_REF/1023;
    float degrees = (voltage*FULL_ANGLE)/GROVE_VCC;
    return degrees;
}
```

Device to Cloud

1. Create Device Sim
2. Run sim and check with Device monitor
3. Modify sim to send real data

Sending message to Cloud

- Send message from device to IoT Hub
- IoT Hub is Microsoft Azure endpoint
- IoT hub always use AMQPS protocol
- Sending messages using Microsoft Azure Service Bus



Create Sim device

1. Create new C# WinForm

2. Add Text box “textBoxD2C” and “textBoxSensorData”

3. NuGet Package Manager install WindowsAzure.ServiceBus

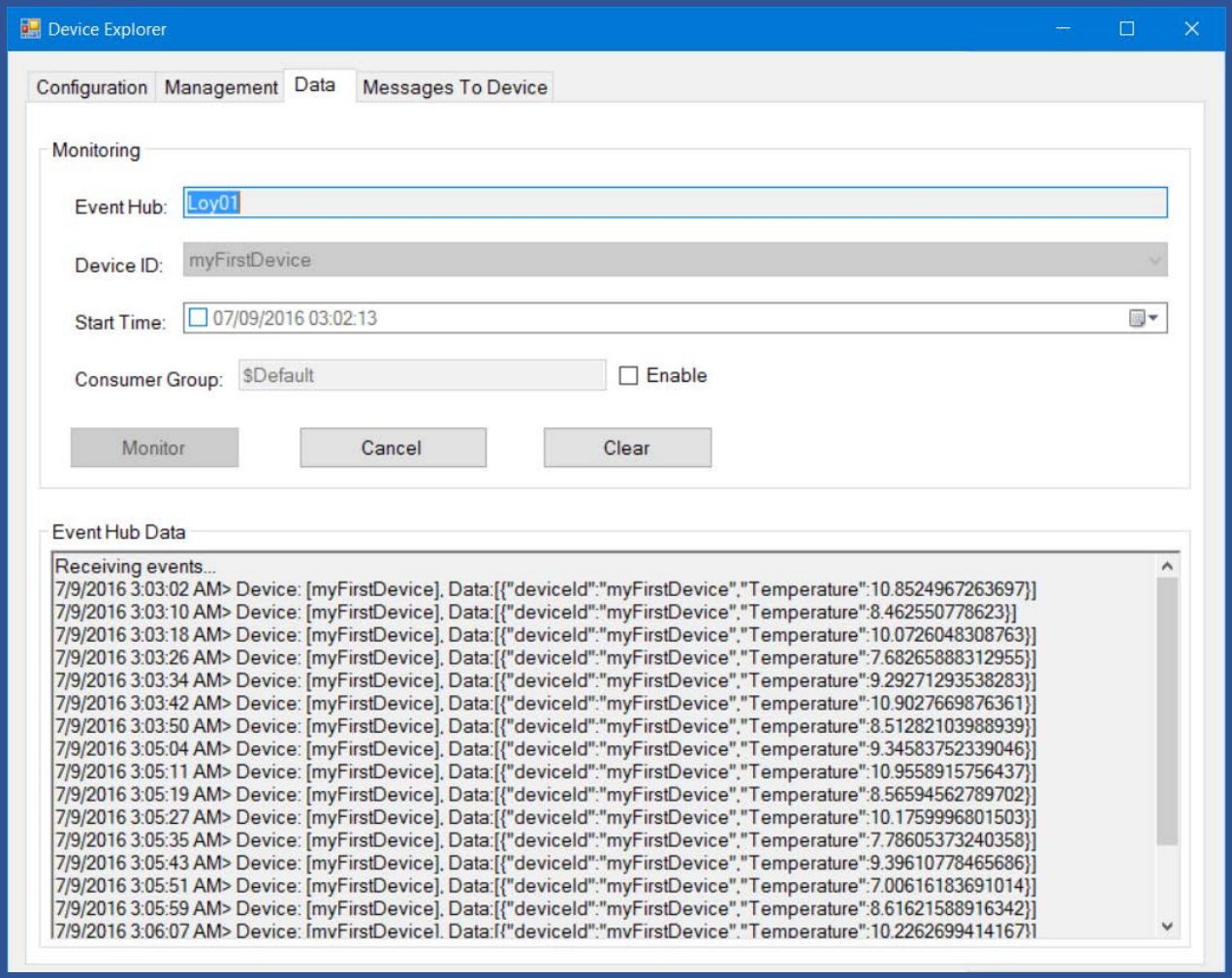
4. Add Code

```
1  using Microsoft.Azure.Devices.Client;
2  using Newtonsoft.Json;
3  using System;
4  using System.Text;
5  using System.Windows.Forms;
```

```
7   namespace Sim01
8   {
        4 references
9       public partial class Form1 : Form
10      {
11          DeviceClient deviceClient;
12          string deviceId = "myFirstDevice";
13          string iotHubUri = "Loy01.azure-devices.net";
14          string deviceKey = "bp4i0ROj+6Z93hmXd64RqH1bmopwT6/2q5762ERROE4=";
            1 reference
15          public Form1()
16          {
17              InitializeComponent();
18              deviceClient =
19                  DeviceClient.Create(iotHubUri,
20                  new DeviceAuthenticationWithRegistrySymmetricKey
21                      (deviceId, deviceKey));
22              Timer myTimer = new Timer();
23              myTimer.Enabled = true;
24              myTimer.Interval = 8000;
25              myTimer.Tick += MyTimer_Tick;
26          }
```

```
27          private void MyTimer_Tick(object sender, EventArgs e)
28          {
29              double temper = 10; // m/s
30              Random rand = new Random();
31              double currentTemper = temper + rand.NextDouble() * 4 - 3;
32              string temperStr = currentTemper.ToString();
33              textBoxSensorData.Invoke(new Action(() =>
34                  { textBoxSensorData.AppendText(temperStr + "\r\n"); }));
35              SendDeviceToCloudMessagesAsync(temperStr);
36          }
            1 reference
37          private async void SendDeviceToCloudMessagesAsync(string data)
38          {
39              var telemetryDataPoint = new
40              {
41                  deviceId = deviceId,
42                  Temperature = Convert.ToDouble(data)
43              };
44              var messageString = JsonConvert.SerializeObject(telemetryDataPoint);
45              var message = new Microsoft.Azure.Devices.Client.Message
46                  (Encoding.ASCII.GetBytes(messageString));
47              await deviceClient.SendEventAsync(message);
48              textBoxD2C.Invoke(new Action(() =>
49                  { textBoxD2C.AppendText(DateTime.Now + messageString + "\r\n"); }));
50          }
51      }
52  }
```

5. Run program
6. Open Device Explorer / Data / Monitor



Create real D2C C# Winapp

1. Copy sim solution
2. Add reference to LoySerialPortUserControl
3. Add LoySerialPortUserControl to Form1
4. Remove Timer and edit Form1 code

```
15          public Form1()
16          {
17              InitializeComponent();
18              deviceClient =
19                  DeviceClient.Create(iotHubUri,
20                  new DeviceAuthenticationWithRegistrySymmetricKey
21                      (deviceId, deviceKey));
22              loySerialPortUc1.OnDataReceived += LoySerialPortUc1_OnDataReceived;
23          }
```

5. Edit code in LoySerialPortUc1_OnDataReceived

```
24          private void LoySerialPortUc1_OnDataReceived(string recieveString)
25          {
26              textBoxSensorData.Invoke(new Action(() =>
27              { textBoxSensorData.AppendText(recieveString + "\r\n"); }));
28              SendDeviceToCloudMessagesAsync(recieveString);
29          }
```

6. Build Temperature hardware
7. Write Arduino Temperature program, delay =1000
8. Upload Arduino code
9. Run real D2C Winapp
10. Watch result in device monitor

D2C Lab

1. Send Ultra Sonic data to Cloud

2. Send Sim and Real data to Cloud concurrently