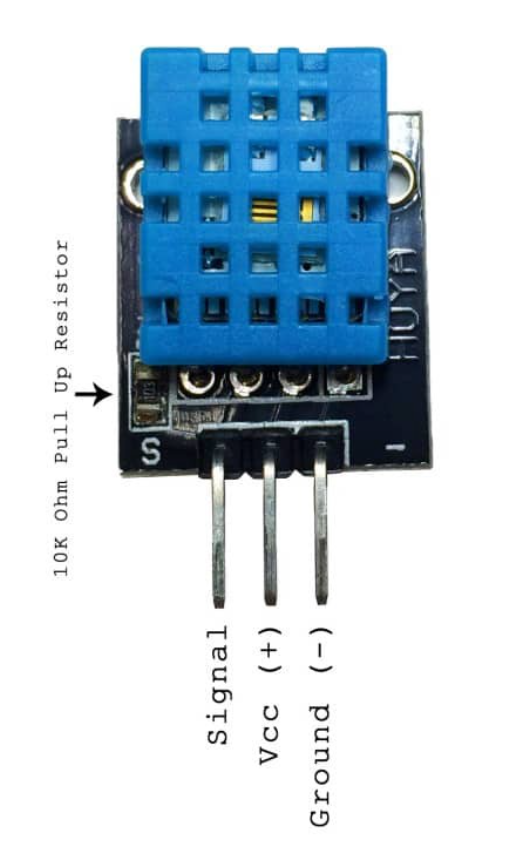
Sensors

DHT11

* DHT11 sensor measures and provides humidity and temperature values serially over a single wire.
* It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.

DHT11 datasheet: <https://www.circuitbasics.com/wp-content/uploads/2015/11/DHT11-Datasheet.pdf>



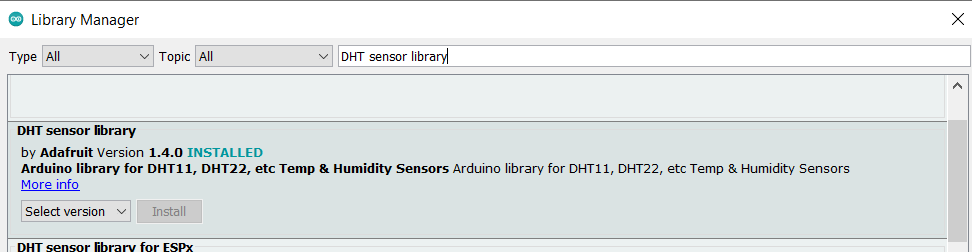
Connection:

Signal can be connected to any digital pin in the Arduino this can be set in the program, pulses are sent from the sensor to communicate the temperature and humidity values.

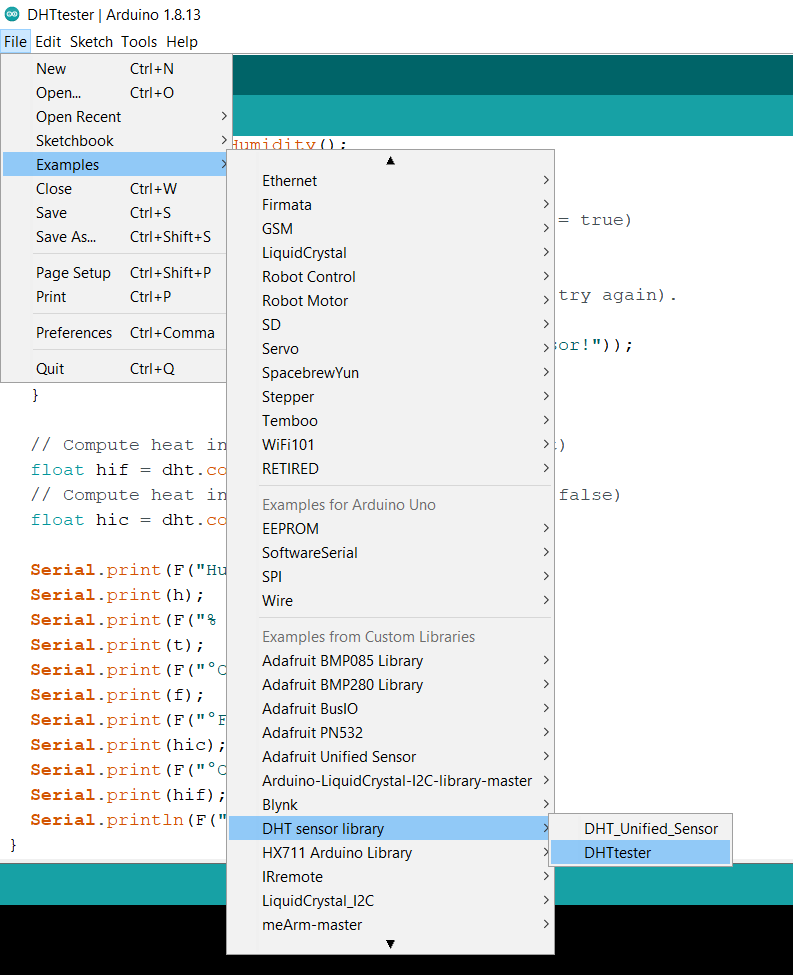
+ to 5V pin of the Arduino

* To Arduino GND

Install the dht11 sensor library by adafruit.



Go to File>Examples>DHT Sensor Library> Open DHT Tester



The code can be modified slightly for our sensor

#include "DHT.h"

#define DHTPIN 2 // Digital pin connected to the DHT sensor

#define DHTTYPE DHT11 // DHT 11

// Initialize DHT sensor.

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); //This starts serial communication, so that the Arduino can send out commands through the USB connection(or bluetooth). The value 9600 is called the 'baud rate' of the connection. This is how fast the data is to be sent. You can change this to a higher value, but you will also have to change the Arduio Serial monitor to the same value.

Serial.println(F("DHTxx test!")); //Serial.print can be used to send text from the arduino to a computer. The differnce between print and println is that println prints a newline at the end

//F() is used to store the text in flash memory rather than RAM. The arduino Nano has 32KB of Flash memory and 2KB o fRAM

dht.begin(); //Begin communication with the sensor

}

void loop() {

// Wait a few seconds between measurements.

delay(2000);

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t) || isnan(f)) {

Serial.println(F("Failed to read from DHT sensor!"));

return;

}

// Compute heat index in Fahrenheit (the default)

float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity: "));

Serial.print(h);

Serial.print(F("% Temperature: "));

Serial.print(t);

Serial.print(F("°C "));

Serial.print(f);

Serial.print(F("°F Heat index: "));

Serial.print(hic);

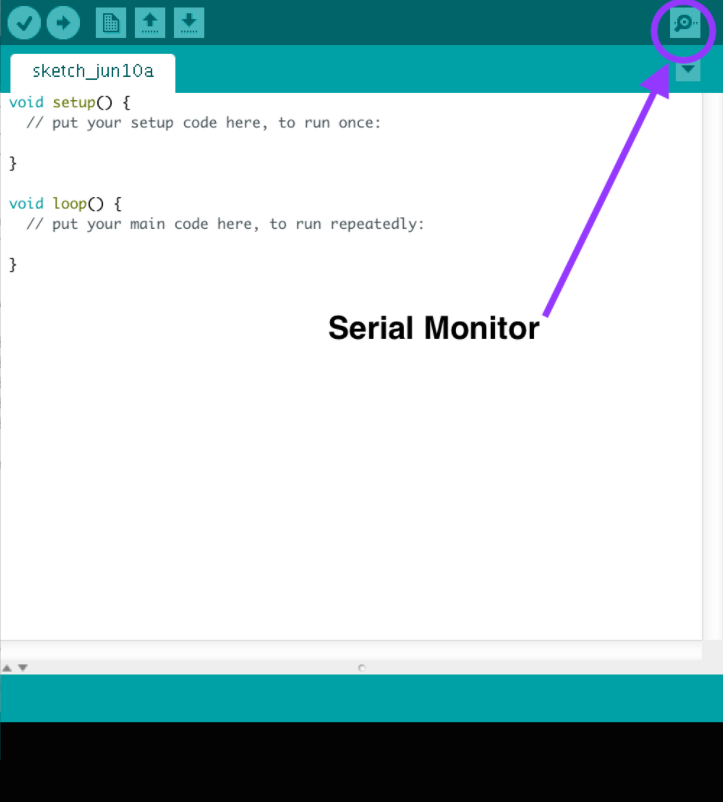
Serial.print(F("°C "));

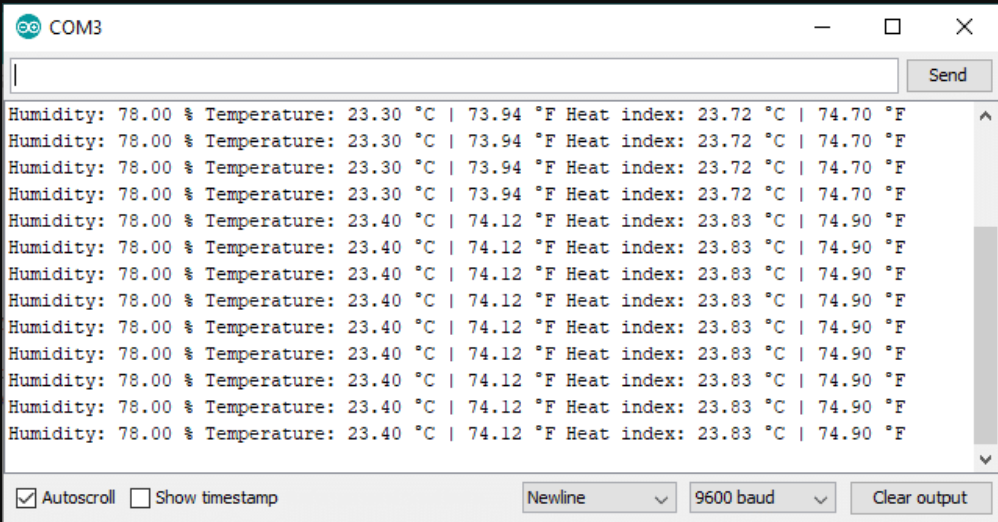
Serial.print(hif);

Serial.println(F("°F"));

}

Upload the code and go to the Serial monitor





Capacitive Soil Moisture Sensor

The soil moisture sensor measures changes in soil by capacitive sensing (changes in capacitance value). The sensor is made of corrosion resistive material.

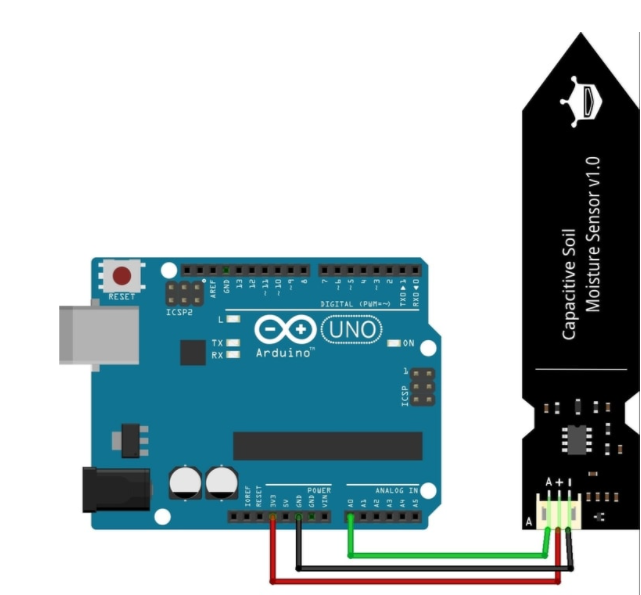
Operating Voltage: 3.3 ~ 5.5 V

## Connection

The analog output can be connected to any analog pin of the Arduino

Vcc of the moisture sensor to Arduino 5V

Gnd of the moisture sensor to Arduino Gnd



Program

void setup() {

Serial.begin(9600); // open serial port, set the baud rate as 9600 bps

}

void loop() {

int val;

val = analogRead(0); //connect sensor to Analog 0

Serial.println(val); //print the value to serial port

delay(100);

}

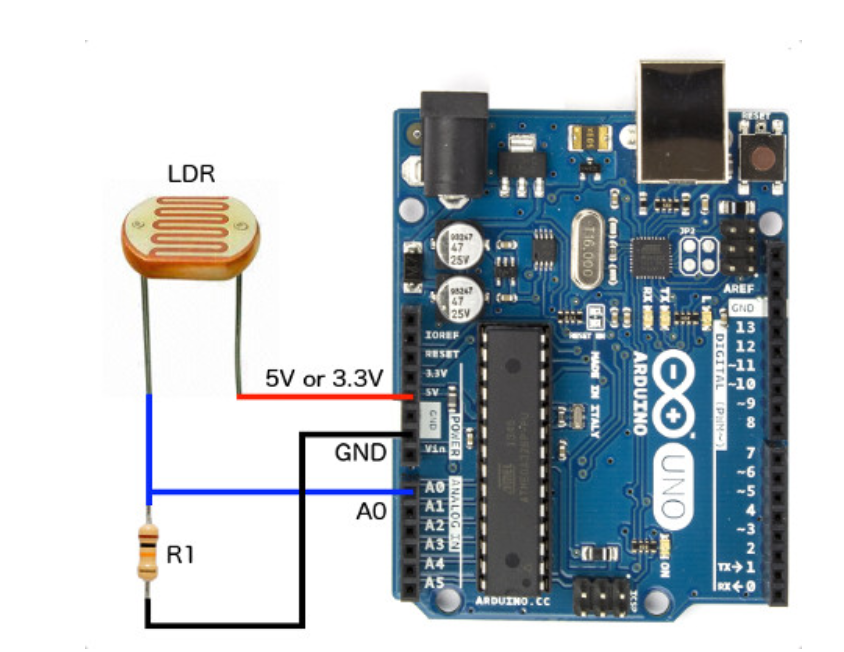
After uploading the code, dip the sensor in water or wet soil. Open the Serial monitor window and observe the analog value change as the sensor is exposed to more moisture.

LDR (Light Dependant Resistor)

The light dependant resistor works on the principle that its resistance value depends on the amount of light it is exposed to.

Connection:

The LDR can be connected to the Arduino in a Resistor Voltage divider configuration as shown in the figure below. The analog voltage received by the Arduino will be in the range 0 to the supply voltage(5V).



Program

The same program as the moisture sensor can be used for the LDR

void setup() {

Serial.begin(9600); // open serial port, set the baud rate as 9600 bps

}

void loop() {

int val;

val = analogRead(0); //connect sensor to Analog 0

Serial.println(val); //print the value to serial port

delay(100);

}

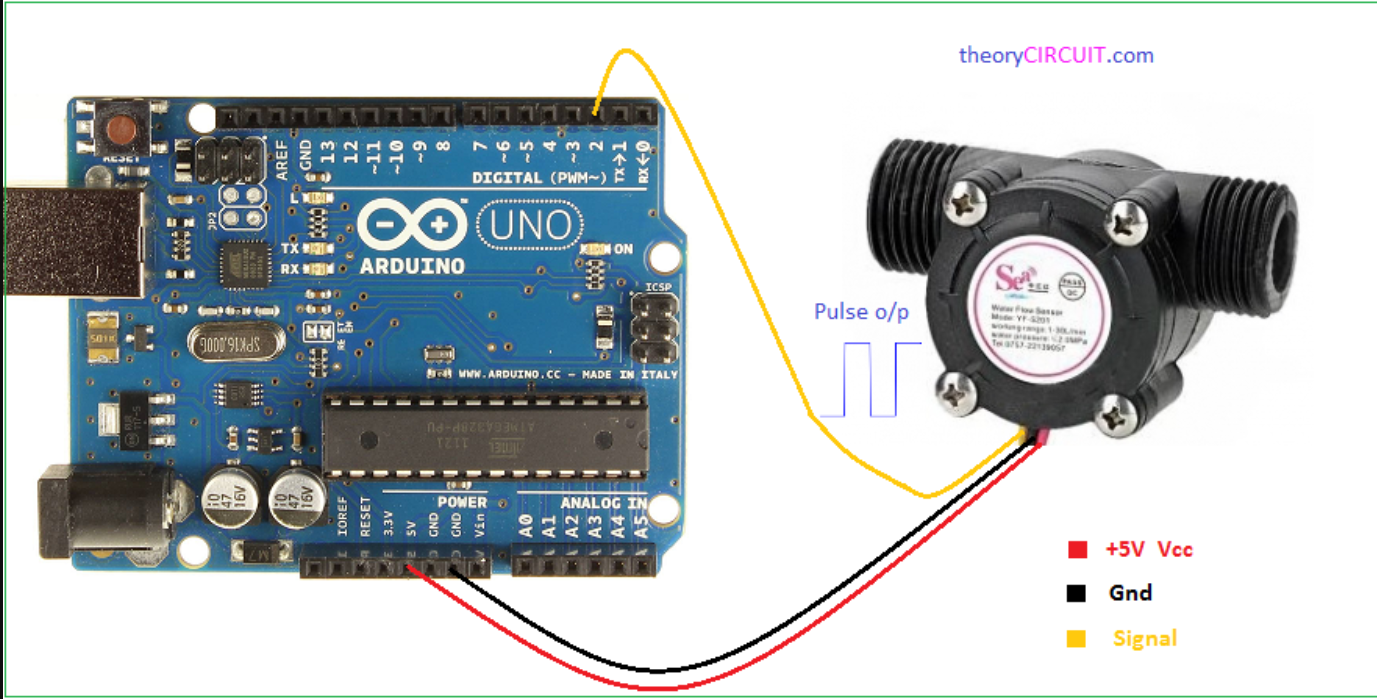
Flow Rate Sensor ( [YF-S402](https://www.seeedstudio.com/M11-1-25-Water-Flow-Sensor-p-1345.html))

The flow rate sensor can be used to measure the volume of water passing per unit time. The flow rate sensor consists of a turbine wheel which rotates as water flows through the sensor, there is a Hall Effect sensor which gives sends out pulses as the turbine wheel rotates. The turbine wheel consists of a magnet, as the wheel rotates the magnet also rotates this causes pulses to be generated by the hall effect sensor. The larger amount of water that flows, the faster the turbine wheel rotates and greater number of pulses are generated by the Hall Effect sensor.

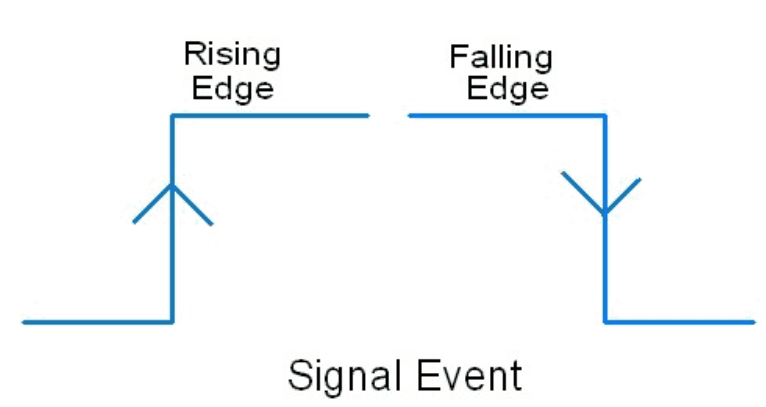
<https://www.seeedstudio.com/blog/2020/05/11/how-to-use-water-flow-sensor-with-arduino/>

Connection:

The Red and Black wires can be connected to Arduino 5V and Gnd respectively. The yellow wire is connected to digital pin 2. Digital pins 2 and 3 can be used as external interrupts. The flow rate sensor is attached to the external interrupt pin as any pulse generated can be counted real time. Using an if statement to check if the pin is high or low would have much more delays and all the pulses may not be counted. <https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>



We can cut the pulses by either counting the rising edge (Low to High) or the Falling edge (High to Low)



Program

#define flowsensorpin 2

volatile double volume = 0;

void flow(){

volume += 1000.0/5880.0;

}

void setup() {

Serial.begin(9600);

pinMode(flowsensorpin, INPUT);

attachInterrupt(digitalPinToInterrupt(flowsensorpin), flow, RISING);

}

void loop() {

Serial.println(int(volume));

delay(100);

}

Explaination:

#define flowsensorpin 2

#define is a useful C++ component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don’t take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time. https://www.arduino.cc/reference/en/language/structure/further-syntax/define/

volatile double volume = 0;

Declaring a variable volatile is a directive to the compiler. It directs the compiler to load the variable from RAM and not from a storage register, which is a temporary memory location where program variables are stored and manipulated. Under certain conditions, the value for a variable stored in registers can be inaccurate. In the Arduino, the only place that volatile is likely to be used is in sections of code associated with interrupts, called an interrupt service routine.

<https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/volatile/>

void flow(){

volume += 1000.0/5880.0;

}

This is the interrupt service routine, the following lines of code are executed when a pulse is detected. When 5880 pulses are detected that means 1 litre of water has passed through the sensor. Thus dividing 1000(millilitre) by 5880 gives the amount of water poured (in mL) for a single pulse. Thus whenever a pulse is detected we add the amount of water poured per pulse to the volume.

attachInterrupt(digitalPinToInterrupt(flowsensorpin), flow, RISING);

digital pin 2 is attached to interrupt 0, we can use the digitalPinToInterrupt() function to convert the pin 2 to the correct interrupt. Here “flow” is the name of the ISR(Interrupt service routine) the lines of code executed when a pulse is detected and we detect the RISING edges of the pulse.

Serial.println(int(volume));

Finally we print the volume as an integer.

submersible Water Pump

The submersible pump consists of a dc motor which is used to pump the water. Our submersible pump has a working voltage range between 3 to 6 V. A dry run of the pump might damage it as this could heat up the motor.



Connection

The pump can draw a maximum current of 220mA, but each digital pin of the Arduino can output around 10mA of current, thus we need to use a transistor turn On and off the pump.

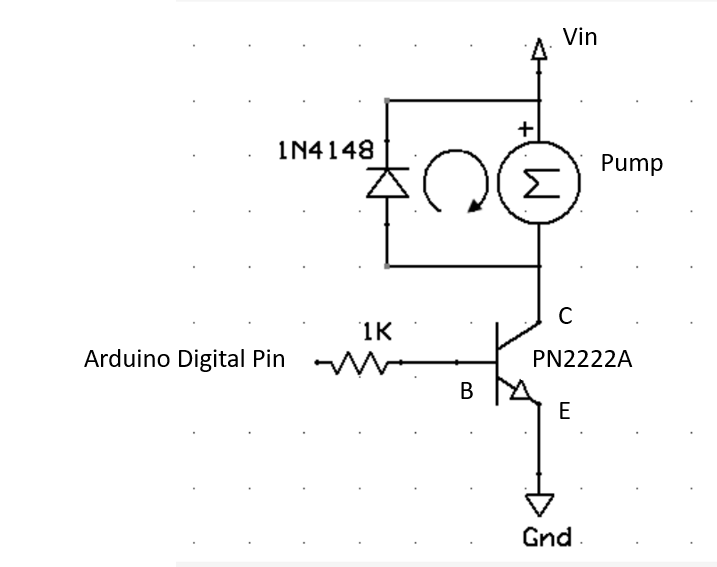
We can use the PN2222A BJT which has a maximum collector current upto 1A. <https://datasheet.datasheetarchive.com/originals/distributors/Datasheets-45/DSA-13238.pdf>

The pump is connected to the Vin pin of the Arduino as this means that the pump will directly receive power from the battery or wall adapter whenever it is switched on by the transistor.

A diode is used to protect the transistor from inductive voltage spikes.

https://en.wikipedia.org/wiki/Flyback\_diode

A 1Kohm resistor is used to limit the base current.



Code

#define pumppin 8 //base of the transistor is connected to digital pin 8 of the Arduino

void setup() {

pinMode(pumppin, OUTPUT);

}

void loop() { //We turn On the pump for 2 seconds, switch it Off and then turn it On again after 10 seconds

digitalWrite(pumppin, HIGH);

delay(2000);

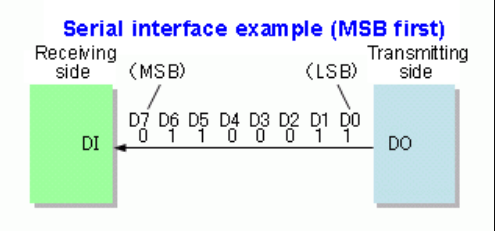
digitalWrite(pumppin, LOW);

delay(10000);

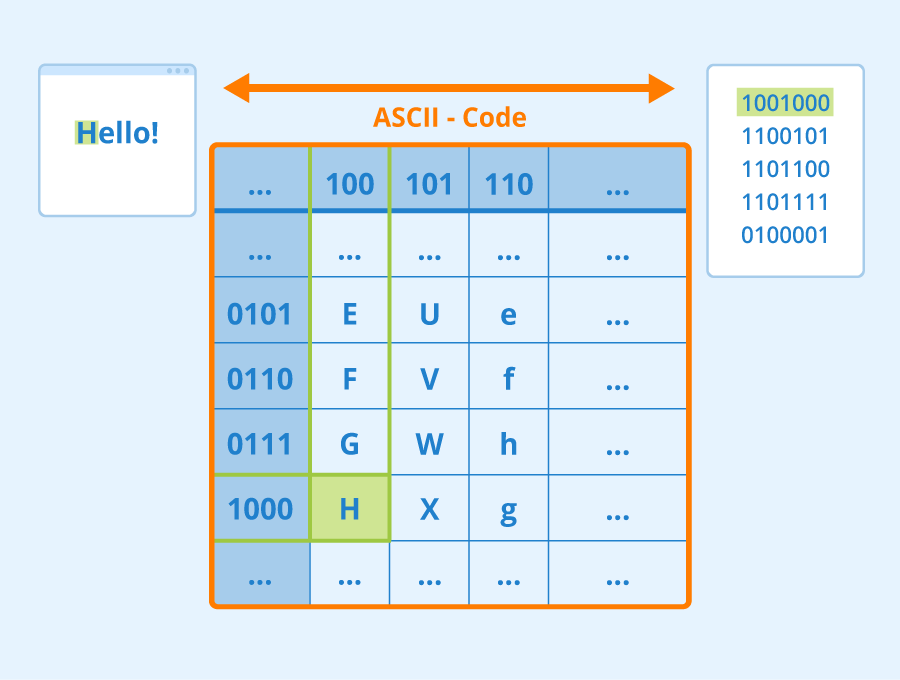
}

Bluetooth HC06

The Bluetooth module HC06 can be used send data wirelessly from the Arduino to your computer. It uses Serial communication to send and receive data. Serial communication involves sending and receiving HIGH /LOW pulses one bit at a time, eight such pulses will constitute a byte.



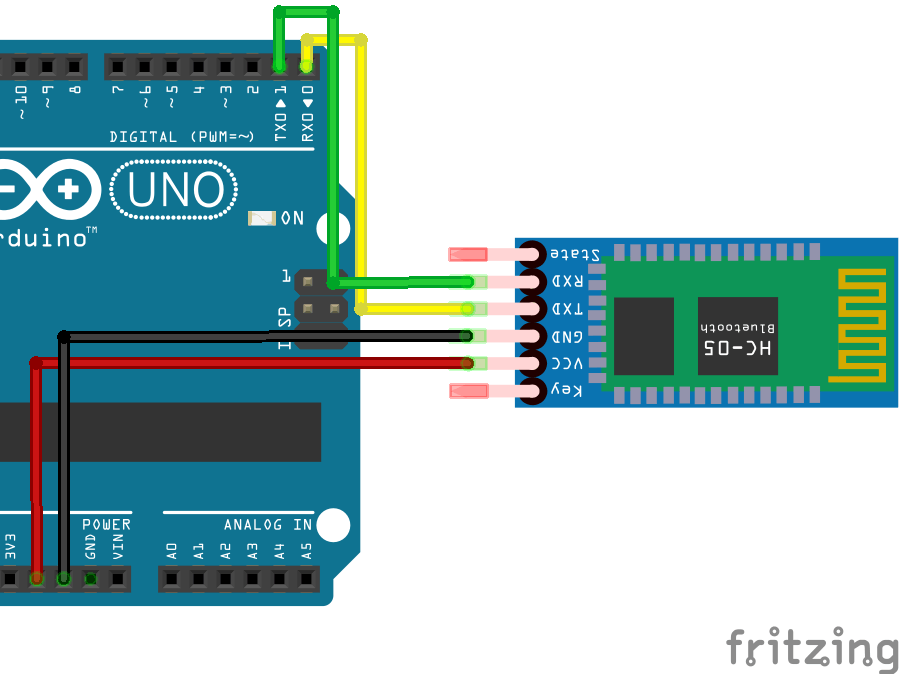
<https://commons.wikimedia.org/wiki/File:Parallel_and_Serial_Transmission.gif>



These pulses/bits can be decoded with the standard ASCII encoding codes, where each character has a binary representation. We will be also be using UTF-8 encoding standard as it supports more characters than ASCII.

Connection:

Connect the Arduino 5V and Gnd to Bluetooth Vcc and Gnd. The transmit pin of Arduino is connected to Recive of Bluetooth and Receive pin of Arduino is connected to Transmit of Bluetooth.



Serial communication is also used to upload code to Arduino, thus we need to disconnect the Tx and Rx of the Arduino from the Bluetooth while uploading code. Thus now our Arduino can communicate Serialy by either Bluetooth or by USB.

Code:

Let us write a code to send humidity values over Bluetooth to the computer from the Arduino. The Arduino should only transmit the Bluetooth value when it has received the character ‘h’.

#include "DHT.h"

#define humiditysensorpin 7

DHT dht(humiditysensorpin, DHT11);

int humiditysensorval = 0;

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

if(Serial.available() > 0){ //checks if the number of bytes received by the Serial buffer is greater than0

userInput = Serial.read(); //Read the character that was received

if(userInput == 'h'){ //if the character received is ‘h’

humiditysensorval = dht.readHumidity();

Serial.print('h'); /\*Let us print h at the beginning of the humidity value to differentiate from other sensor values\*/

Serial.println(humiditysensorval); //print the sensor vale

Serial.flush(); /\*flush() waits until the entire data has been sent serially, this is to ensure that the whole data has been sent successfully\*/

}

}

}

Notice that the above lines of code are similar to the codes we have used to print sensor data on to the Serial Monitor. This is because transmitting data over USB and Bluetooth from the Arduino to the Computer both use Serial communication. Thus the above lines of code would if used either a Bluetooth module or connect via a USB cable. To test out the code by USB cable instead of Bluetooth you send the character ‘h’ via the Serial monitor and see the humidity value.

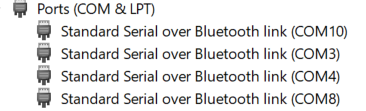
Now we can connect the humidity sensor and the Bluetooth module to the Arduino.

First we have to connect to the Bluetooth Module:

Windows

1. Go to the Bluetooth settings and pair with the Bluetooth HC06
2. Go to device manager and check all the COM ports available for Bluetooth

Communication



Raspberry Pi

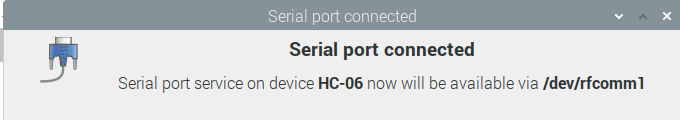
1. Go to terminal and run the following commands, the following packages lets us setup the Raspberry Pi’s Bluetooth for Serial Communication.

sudo apt-get update

sudo apt-get install pi-bluetooth  
sudo apt-get install bluetooth bluez blueman

1. Restart the Raspberry Pi and go to menu -> preferences -> bluetooth manager.

Click on the search icon, after HC-06 appears right click on it and then select setup. Pair the device with the default key 1234 or 0000. Go to connect tab and Select “Serial Port”. A dialog will appear



Let us write a python program to receive sensor data from the Arduino over Bluetooth.

import serial

import time

#Create a Serial object

ser = serial.Serial('COM10',baudrate = 9600, timeout=3) #For Windows, if COM10 does not work try #all the other Bluetooth COM ports

#ser = serial.Serial('/dev/rfcomm0',baudrate = 9600, timeout=3) #For Raspberry Pi

#The timeout is set to 3 seconds, thus we will wait 3 seconds to read data. This set so that we don’t #wait indefinitely

time.sleep(3) #Wait 3 seconds for the buetooth to connect

while True:

userInput = input('\nGet data point?') #ask the user to enter a character to request humidity value

if userInput == 'h':

ser.write(b'h') #Send ‘h’ over bluetooth

humidityval = ser.readline().decode('utf-8') #read the humidity value decode using utf-8

if len(humidityval) > 0: #check if we have received any data

if humidityval[0] == 'h': #check if the first character is ’h’ for humidity

humidityval = int(humidityval[1:]) #slice the list to remove ‘h’ and convert to int

print("Humidity sensor val: ",humidityval)

Run the above code and type h in the shell window to view the humidity sensor value.