Cyber Physical Systems

Basics of Arduino

Sambit Sahoo

sambit22@iiserb.ac.in

May 23, 2025

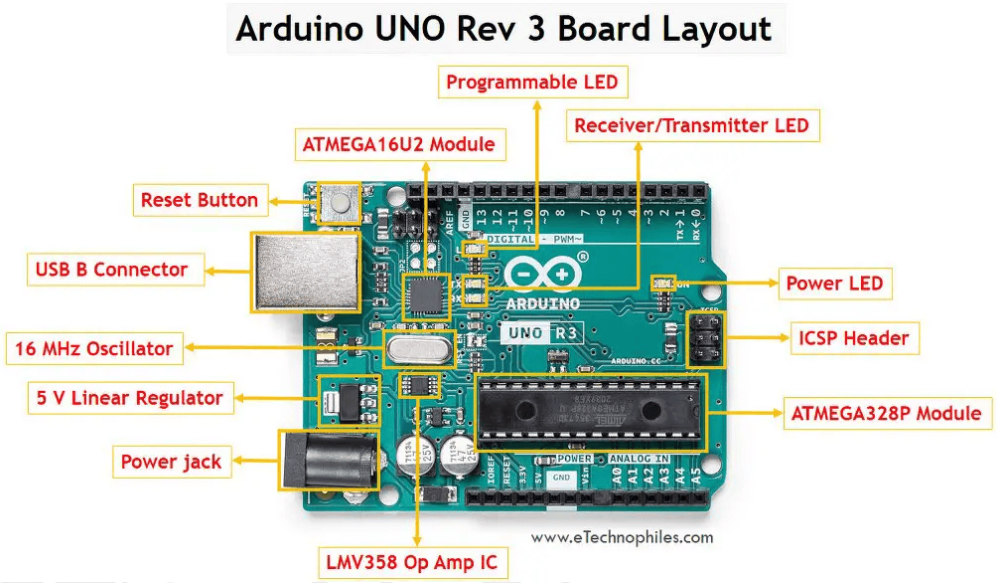
Table of Contents

Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Parts of the Arduino UNO Board

- Microcontroller
- Digital I/O Pins (0–13)
- Analog Input Pins (A0–A5)
- Power Pins
- USB Port
- Voltage Regulator
- Crystal Oscillator (16 MHz)
- Reset Button
- Power LED (ON)
- TX/RX LEDs
- ICSP Header (In-Circuit Serial Programming)
- Bootloader



Parts of the Arduino UNO Board

Microcontroller

- The brain of the Arduino.
- Executes the program (sketch) uploaded from your computer.
- Contains memory (Flash, SRAM, EEPROM), clock, timers, and I/O pins.

Digital I/O Pins

- Can be used to read or send digital signals (HIGH/LOW).
- Pins 0 and 1 are also used for serial communication (RX/TX).

Analog Input Pins (A0–A5)

- Used to read analog sensors (e.g., temperature sensors, potentiometers).
- Convert analog voltage (0–5V) to digital values (0–1023) using an ADC.

Parts of the Arduino UNO Board (contd..)

Power Pins

- **Vin**: Input voltage if using an external power source.
- **5V**: Regulated 5V supply from USB or onboard regulator.
- 3.3V output (lower power sensors).
- **GND (Ground)**: Common ground reference.

USB Port

- Used to connect to a computer for Uploading code and Serial communication.
- Also powers the board when connected.

Voltage Regulator

- Regulates voltage down to a stable 5V or 3.3V to protect components.
- Works when external power is used (e.g., battery or adapter).

Parts of the Arduino UNO Board (contd..)

Crystal Oscillator (16 MHz)

- Provides the clock signal that keeps the microcontroller running at a consistent speed.

Reset Button

- Resets the microcontroller.
- Useful for restarting the program from the beginning without unplugging the board.

Power LED

- Lights up when the board is powered.

TX/RX LEDs

- Indicate data transmission (TX) and reception (RX) over USB or serial.

Parts of the Arduino UNO Board (contd..)

ICSP Header (In-Circuit Serial Programming)

- Used to program the microcontroller directly using an external programmer.
- Bypasses the bootloader

Bootloader

- Pre-installed firmware that allows uploading code over USB..
- Eliminates the need for a separate hardware programmer.

1. Blinking External LED

Blinking an external LED is often the first project when learning microcontrollers like Arduino. It demonstrates basic digital output control and timing.

- An LED is connected to one of the Arduino's digital pins via a current-limiting resistor.
- The Arduino turns the LED ON and OFF repeatedly using `digitalWrite()`.
- The `delay()` function controls the time intervals between ON and OFF states.
- This project introduces fundamental concepts:
  - Configuring pin modes with `pinMode()`.
  - Writing digital signals.
  - Using delays to create visible blinking.
- It is analogous to "Hello World" in programming — a simple, clear introduction to hardware interaction.

1. Blinking external LED

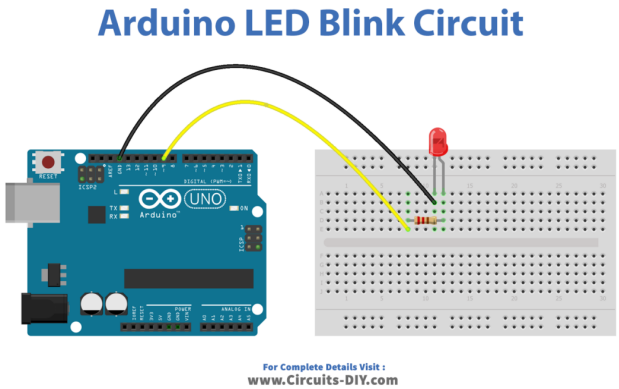
blinking_ext_LED | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Arduino Uno

blinking_ext_LED.ino

```
1   //(Basic LED Blink Example)
2   const int ledPin = 13; // Pin where LED is connected
3
4   void setup() {
5     pinMode(ledPin, OUTPUT); // Set the LED pin as output
6   }
7
8   void loop() {
9     digitalWrite(ledPin, HIGH); // Turn the LED on
10    delay(1000);                // Wait for 1 second
11    digitalWrite(ledPin, LOW);  // Turn the LED off
12    delay(1000);                // Wait for 1 second
13  }
14
15
```



2. Morse Code SOS Flasher

Morse code is a method of encoding textual information using sequences of short and long signals, known as "dots" and "dashes." The SOS signal in Morse code is denoted by:

... — ...

This signal is universally recognized as a distress call. In the Morse Code SOS Flasher experiment using an Arduino:

- An LED is used to flash the SOS pattern.
- The timing of dots (short flashes) and dashes (long flashes) is precisely controlled using code.
- This experiment helps in understanding digital output control and timing functions.

2. Morse Code SOS Flasher


```
1   int ledPin = 13
2   int durations[] = {200,200,200,500,500,500,200,200,200}
3
4   void setup() {
5     pinMode(ledPin, OUTPUT); // put your setup code here, to run once:
6   }
7
8   void flash(int duration) {
9     digitalWrite(ledPin, HIGH);
10    delay(duration);
11    digitalWrite(ledPin, LOW);
12    delay(duration);
13  }
14
15  void loop() {                    // put your main code here, to run repeatedly:
16    for (int i = 0; i<9; i++){
17      flash(durations[i]);
18    }
19
20    delay(1000);                   // wait 1 second before we start
21  }
```

3. Morse Code Translator

A Morse Code Translator converts regular text into Morse code, which uses a combination of dots (.) and dashes (–) to represent letters, digits, and punctuation.

- Each character is mapped to its corresponding Morse code equivalent.
- The Arduino reads input characters (typically from a serial monitor or sensor).
- It then translates each character and outputs Morse code using:
  - LED blinks (short/long flashes),
  - Sound signals (buzzer beeps), or
  - Serial printouts.
- Timing is essential:
  - Dot: 1 time unit
  - Dash: 3 time units
  - Space between parts of the same letter: 1 unit
  - Space between letters: 3 units
  - Space between words: 7 units

3. Morse Code Translator

morsecode_translator | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Arduino Uno

morsecode_translator.ino

```
1   int ledPin = 13;
2
3   char* letters[] = {
4     ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..",    // A-I
5     ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.",  // J-R
6     "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."         // S-Z
7   };
8
9   char* numbers[] = {"-----", ".----", "..---", "...--", "....-", ".....", "-....",
10    "--...", "---..", "----."};
11
12  int dotDelay = 200;
13
14  void setup(){
15    pinMode(ledPin, OUTPUT);
16    Serial.begin(9600);
17  }
18
19  void loop() {               // put your main code here, to run repeatedly:
20    char ch;
21    if (Serial.available()) {              // is there anything to read from USB?
22      char ch = Serial.read();             // read a single character
23
24      if (ch >= 'a' && ch <= 'z') {
25        flashSequence(letters[ch - 'a']);
26      }
27      else if (ch >= 'A' && ch <= 'Z') {
28        flashSequence(letters[ch - 'A']);
29      }
30      else if (ch >= '0' && ch <= '9') {
31        flashSequence(numbers[ch - '0']);
32      }
```

3. Morse Code Translator (contd..)

morsecode_translator | Arduino IDE 2.3.4

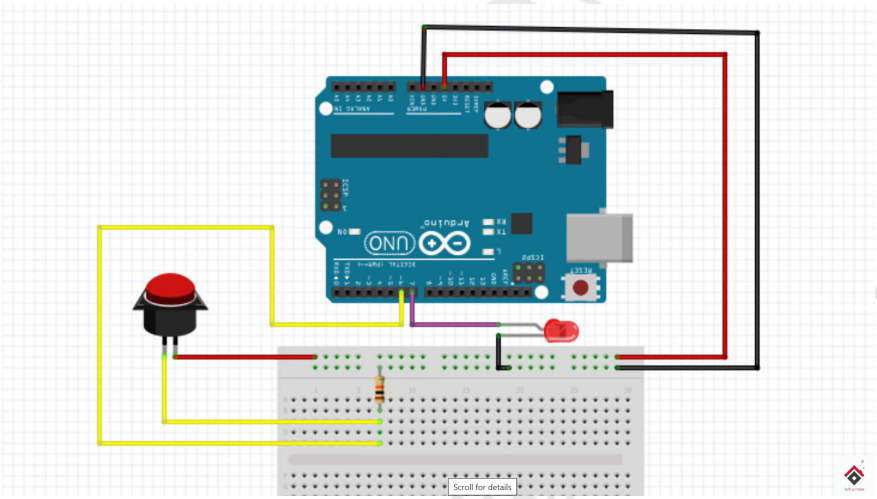
File Edit Sketch Tools Help

Arduino Uno

morsecode_translator.ino

```
    }
    else if (ch >= '0' && ch <= '9') {
      flashSequence(numbers[ch - '0']);
    }
    else if (ch == ' ') {
      delay(dotDelay * 4);               // space between words
    }
  }
}

void flashSequence(char* sequence) {
  int i = 0;
  while (sequence[i] != '\0') {          // loop through each symbol in Morse code
    flashDotOrDash(sequence[i]);
    i++;
  }
  delay(dotDelay * 3);                   // gap between letters
}

void flashDotOrDash(char dotOrDash) {
  digitalWrite(ledPin, HIGH);
  if (dotOrDash == '.') {
    delay(dotDelay);                     // dot = 1 unit
  } else {
    delay(dotDelay * 3);                 // dash = 3 units
  }
  digitalWrite(ledPin, LOW);
  delay(dotDelay);                       // gap between dots and dashes
}

}
```

4. Pushbutton and LED (using digitalRead)

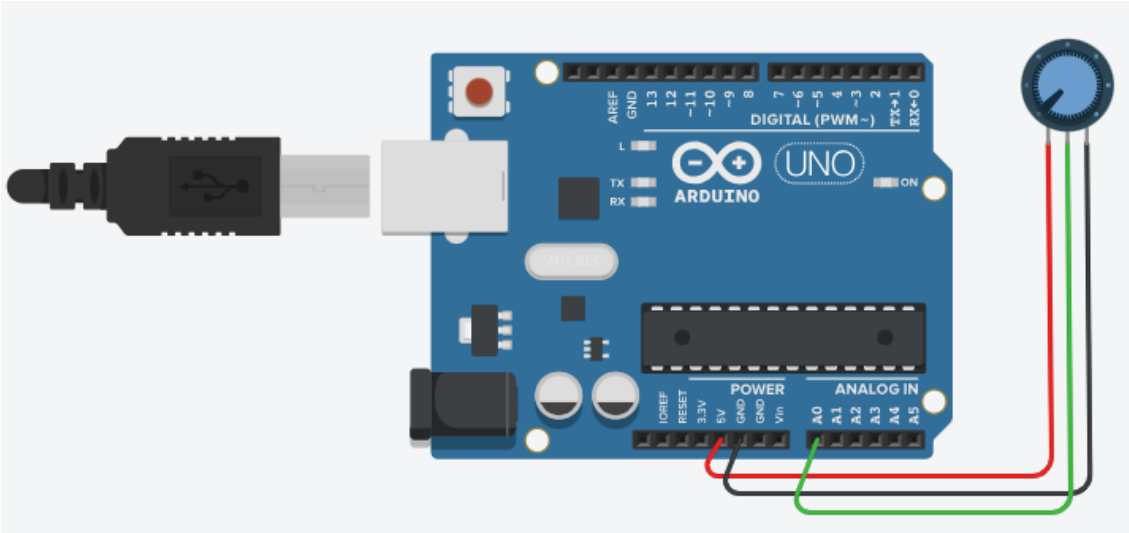


pushbutton_digitalread.ino

```
1   int s = 6;
2   int LED = 7;
3   intx;
4
5   void setup() {
6     serial.begin(9600);
7     pinMdoe(6,INPUT);
8     pinMode(7,OUTPUT);
9   }
10
11  void loop() {
12    x = digitalRead(s);
13    serial.println(x);
14    delay(1000);
15
16    if (x == 0){
17      digitalWrite(7, LOW);
18    }
19
20    else{
21      digitalWrite(7,HIGH);
22    }
```

- Serial is used because Arduino communicates data to the computer via serial communication. To start serial communication we have to set the baud rate (no. of bits transferred /second)

5. Potentiometer (using analogRead)

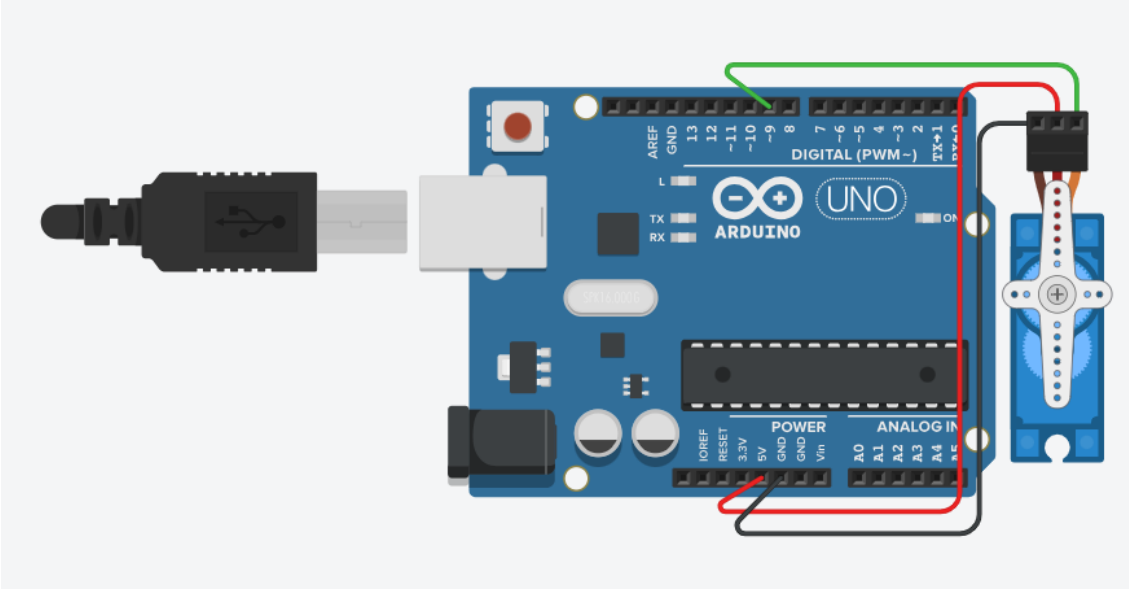


pot.ino

```
1  int pot = A0;
2  int x;
3
4  void setup(){
5    Serial.begin(9600);
6    pinMode(A0, INPUT);
7  }
8
9  void loop(){
10   x = analogRead(pot);
11   Serial.println(x);
12   delay(300);
13 }
```

- Demonstrates how to read analog input using ‘analogRead()‘ from a variable resistor (potentiometer).
- Output value ranges from 0 to 1023, representing voltage between 0V to 5V.

6. Servo Motor with Arduino

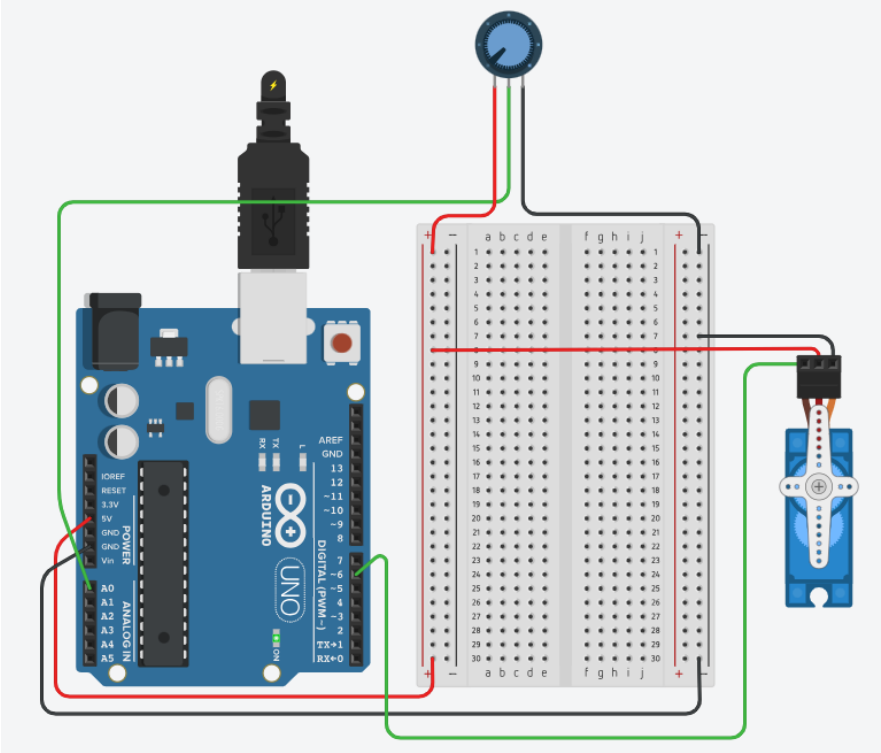


servo_motor.ino

```
1
2   #include <Servo.h>
3   Servo s1;
4
5   void setup(){
6     s1.attach(9);
7   }
8
9   void loop(){
10    for (int i = 0; i<= 180; i++){
11    s1.write(i);
12    delay(15);
13    }
14    for (int i = 180; i>=0; i--){
15    s1.write(i);
16    delay(15);
17    }
18  }
```

- Servo motors are used for precise control of angular position.
- Controlled using PWM signals; the angle is set via ‘servo.write()‘ in Arduino.

7. Control Servo Motor with Potentiometer

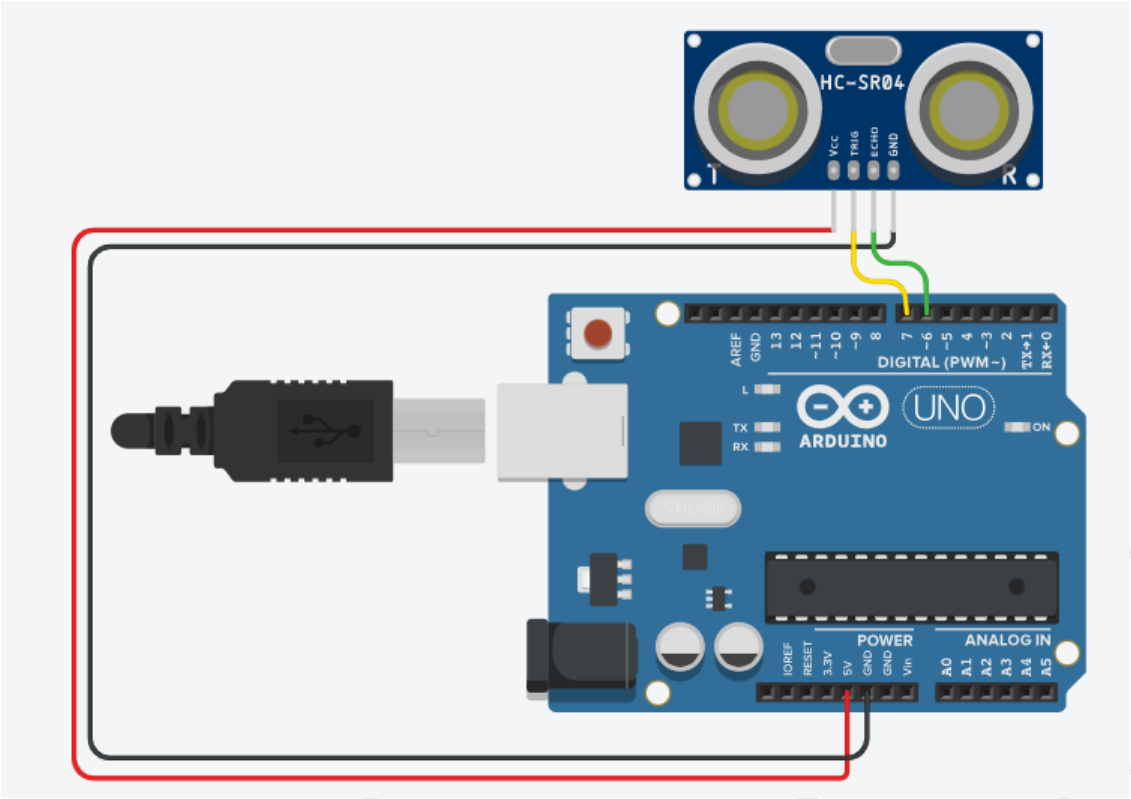


pot_servomotor.ino

```
#include <Servo.h>
int pot = A0;
Servo s1;
int x;
int value;

void setup(){
  Serial.begin(9600);
    s1.attach(6);
    pinMode(A0, INPUT);
}

void loop(){
    x = analogRead(pot);
    value = map(x,0,1023,0,180);
    s1.write(value);

}
```

- The ‘map()‘ function converts the potentiometer’s analog input range (0–1023) to a suitable angle range (e.g., 0–180°) for servo rotation.
- Similar experiment – Potentiometer controlling LED brightness (uses PWM concept).

8. Ultrasonic Sensor HC-SR04 with Arduino



ultrasonic_sensor.ino

```
1  int trig = 7;
2  int echo = 6;
3  int timeInmicro;
4  int distanceIncm;
5
6  void setup(){
7      Serial.begin(9600);
8    pinMode(trig, OUTPUT);
9      pinMode(echo, INPUT);
10 }
11
12 void loop(){
13   digitalWrite(trig, LOW);
14     delay(200);
15     digitalWrite(trig, HIGH);
16     delay(200);
17     digitalWrite(trig, LOW);//pulse generated
18
19     timeInmicro = pulseIn(echo, HIGH);
20     distanceIncm = timeInmicro/29/2;
21     Serial.println(timeInmicro);
22 }
```

- HC-SR04 measures distance using ultrasonic sound waves and echo timing.
- ‘pulseIn()‘ returns the time (µs) the echo pin stays HIGH.
- Distance is calculated using: `distance = time / 29 / 2`.

9. Temp Monitor System using DHT and Motor

temp_monitor_system_dht_motor.ino

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);

int in1 = 2;
int in2 = 3;
int en1 = 6;
int y;

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

  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(en1,OUTPUT);
}

void loop() {
  delay(2000);
  float t = dht.readTemperature();
  //checking if senor is reading inputs
  if (isnan(t)){
    Serial.println("Failed to read from DHT");
    return;
  }

  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  y = map(t,20,50,0,255);
  analogWrite(en1,y);
}
```

- **DHT11 Sensor:** Measures temperature and sends data to Arduino.
- **PWM Control:** Fan speed is adjusted using PWM based on temperature.

9. Temp Monitor System using DHT and Motor

temp_monitor_system_dht_motor_1.ino

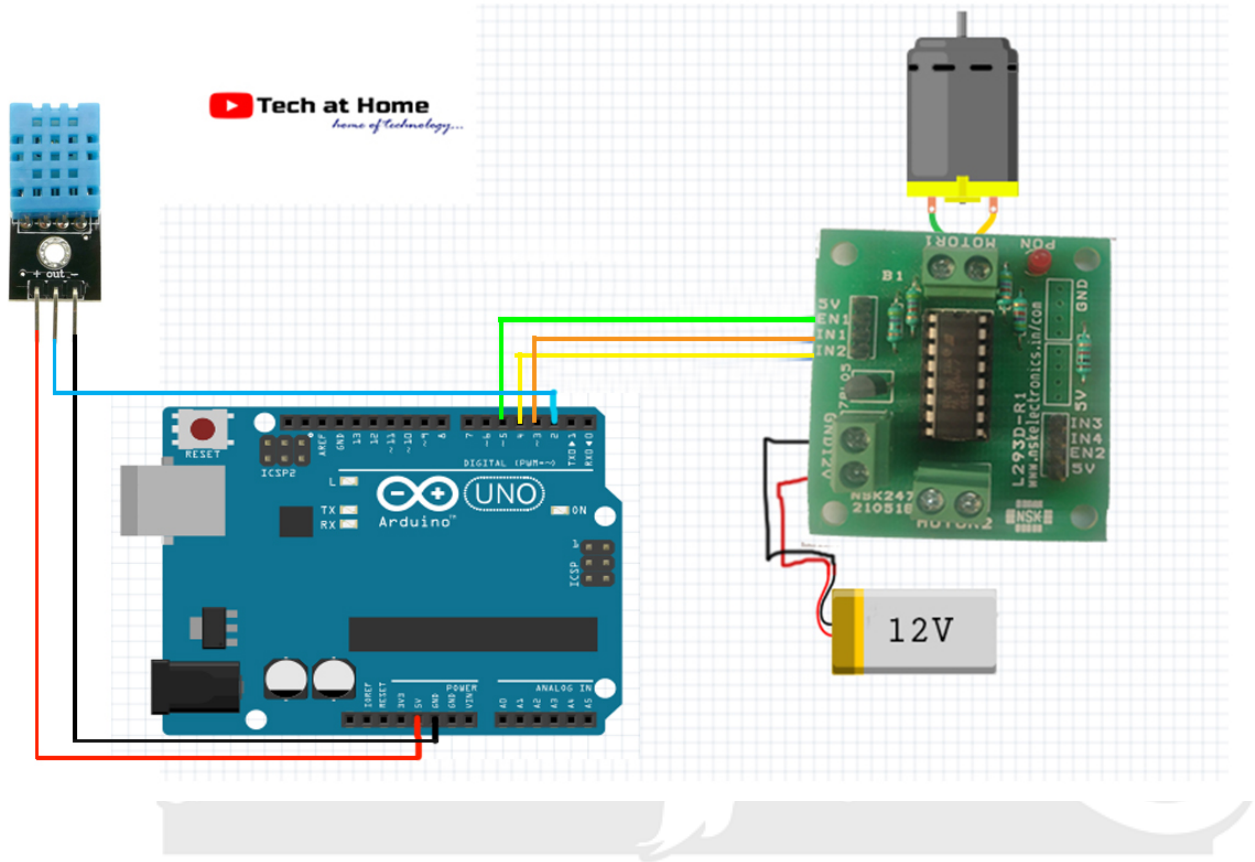
```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);

int in1 = 2;
int in2 = 3;
int en1 = 6;
int y;

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

  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(en1,OUTPUT);
}

```

temp_monitor_system_dht_motor_1.ino

```
void loop() {
  delay(2000);
  float t = dht.readTemperature();
  //checking if senor is reading inputs
  if (isnan(t)){
    Serial.println("Failed to read from DHT");
    return;
  }

  if (t <= 28){
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
    analogWrite(en1,0);
  }

  else if ((t > 28) && (t <= 30)){
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    analogWrite(en1,100);
  }

  else if (t > 30){
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    analogWrite(en1,255);
  }
```

- without using map function. Fan speed is operated by the conditions shown in the code.

9. Temp Monitor System using DHT and Motor

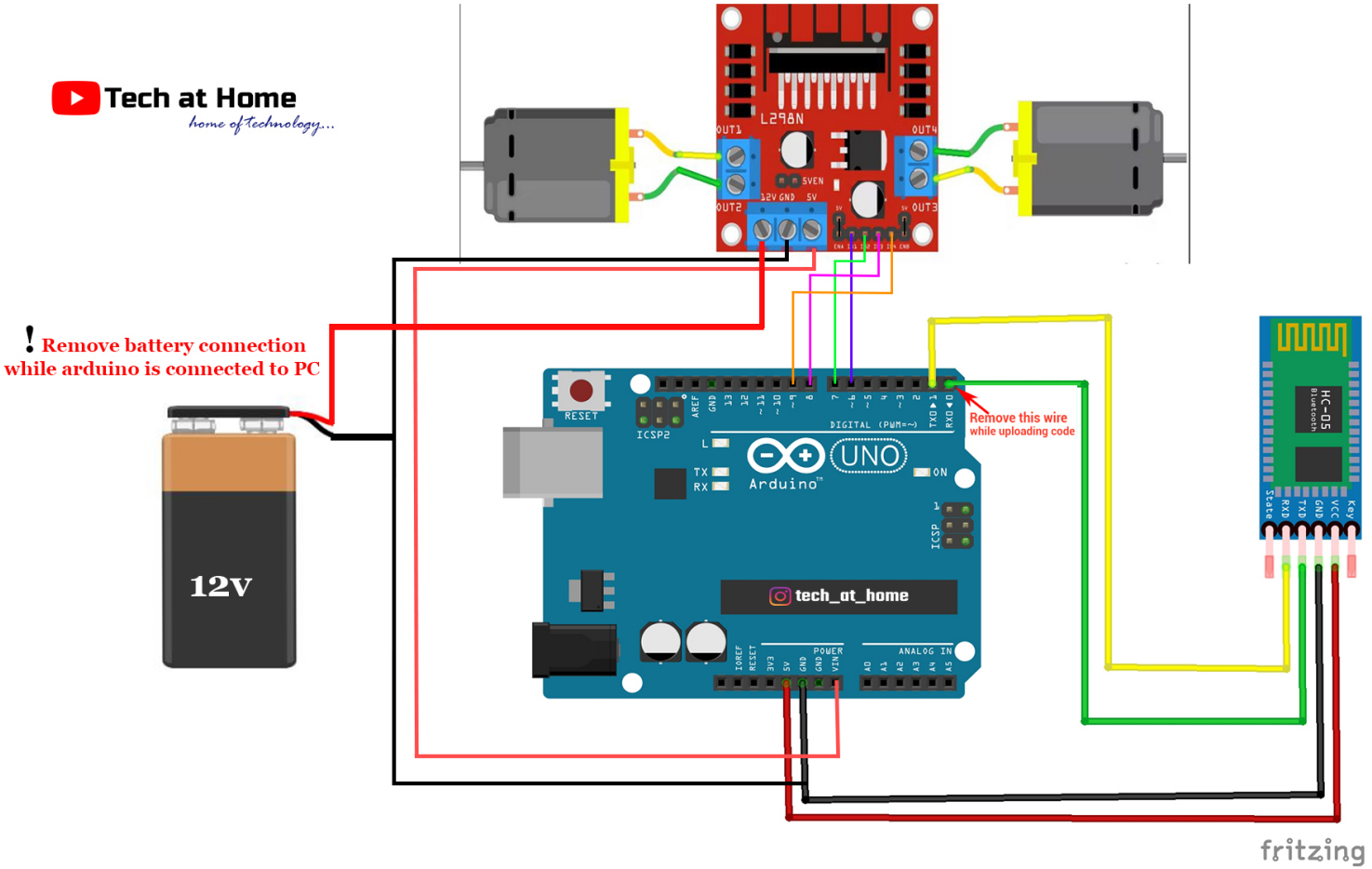


10. Bluetooth Controlled Car

btmodule3_automated_car_l298n.ino

```
int in1 = 6;
int in2 = 7;
int in3 = 8;
int in4 = 9;
int data;

void setup() {
  Serial.begin(9600);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
}

void loop() {
  while (serial.available() > 0){
    data = serial.read();
    Serial.println(data);
  }

  if (data == "F"){
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }

  if (data == "B"){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }

  if (data == "L"){
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
  }

  if (data == "R"){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }
}

```

10. Bluetooth Controlled Car



- Here I learned UART-based serial communication to receive and decode commands from the Bluetooth module.

References

- **YouTube:** Arduino Projects
- **Book:** Simon Monk, "30 Arduino Projects for the Evil Genius"
- **Simulations:** Tinkercad

Thanks.