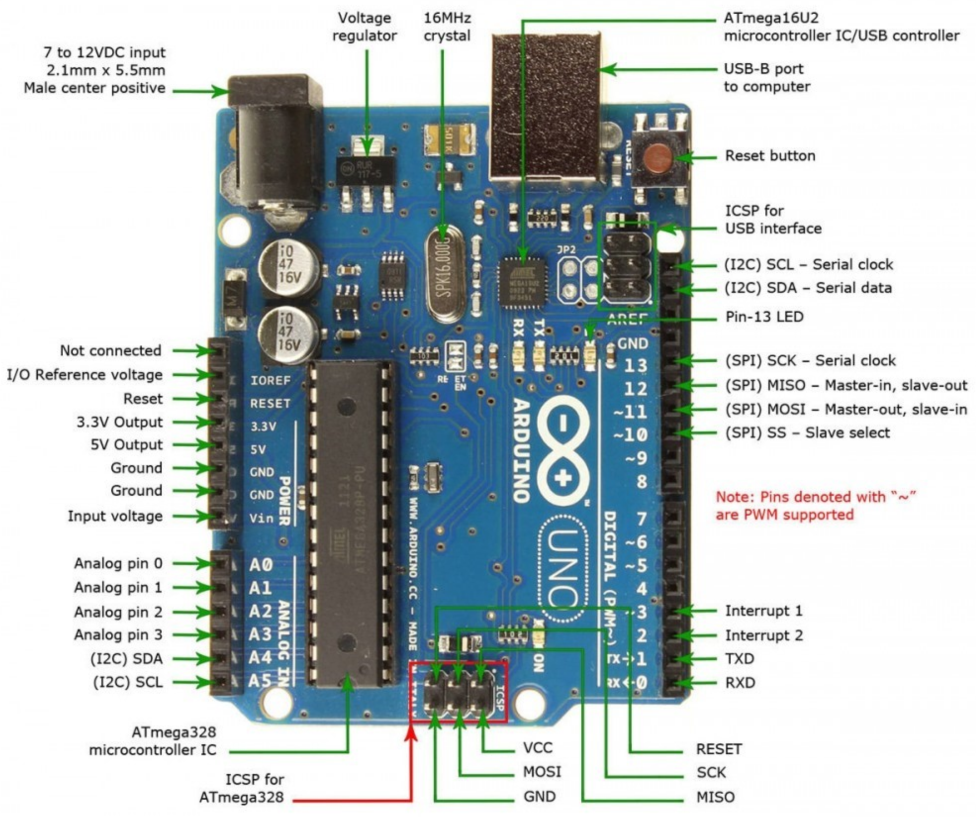
ROBOTIC WORKSHOP 7.0

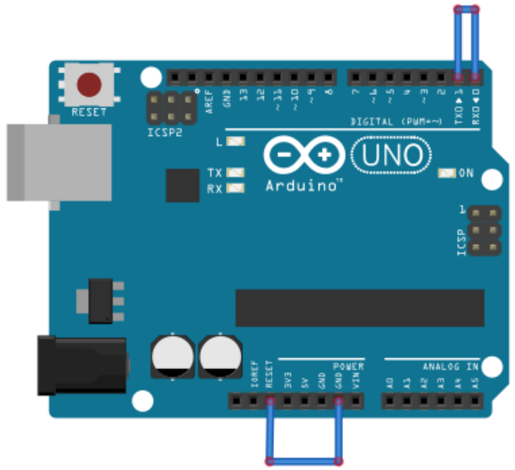
MODULE 1

1.1 INTRODUCTION TO ARDUINO

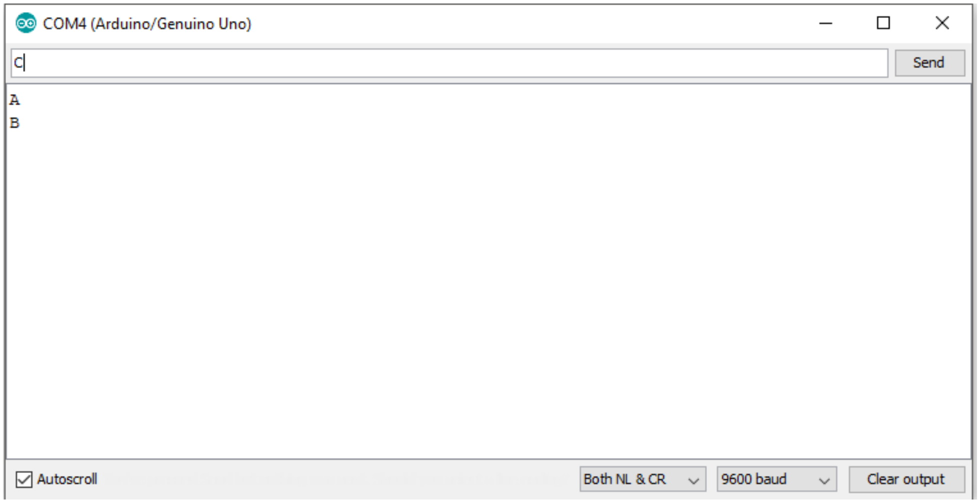
Arduino is a platform and an environment, not just a specific product it is a collection of open-source microcontroller boards, which contain small pieces of code, called the Arduino bootloader. This code allow us to integrate with the Arduino IDE which provides a set of libraries. Arduino specific libraries designed to replace the more complex intricacies of microcontroller programming with easy-to-use functions and methods.



1.2 SERIAL MONITOR – LOOPBACK TEST



Open Arduino IDE. Start serial monitor after selecting your port and send data by typing. Whatever you write should be echoed back.



COM4 (Arduino/Genuino Uno)

C

```
A
B
```

☑ Autoscroll

Both NL & CR

9600 baud

1.3 SERIAL PRINT

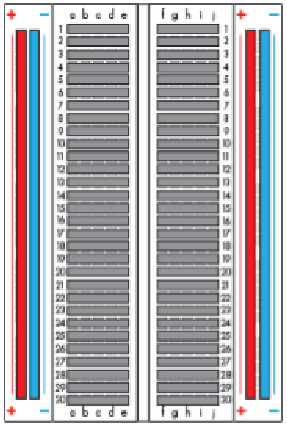
```
// ----------------- SERIAL PRINT -------------------- //

int count;

void setup() {
  // put your setup code here, to run once:
  Serial.begin (9600);  // open serial port to send data
                        // back to the computer at 9600 bps
  Serial.println ("This code run once");
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print ("This code run repeatedly");
  Serial.println (count);
  count++;
}
```

1.4 READING INPUT FROM SERIAL MONITOR

```
// ---------------- READ SERIAL CHAR ------------------ //

void setup() {
  Serial.begin (9600);  // open serial port to send data
                        // back to the computer at 9600 bps
}

void loop() {
  //if there is input from serial monitor
  if (Serial.available()){
    //read input character from serial monitor
    char inputChar = Serial.read();
    Serial.println(inputChar);
    if (inputChar == '1')
      Serial.println ("Character 1 has been entered");
  }
}
```

```
// ---------------- READ SERIAL INT ------------------- //

void setup() {
  Serial.begin (9600);  // open serial port to send data
                        // back to the computer at 9600 bps
}

void loop() {
  //if there is input from serial monitor
  if (Serial.available()){
    //read input character from serial monitor
    int inputInt = Serial.parseInt();
    Serial.println(inputInt);
    if (inputInt == 1)
      Serial.println ("Integer 1 has been entered");
  }
}
```

1.5 BREADBOARD

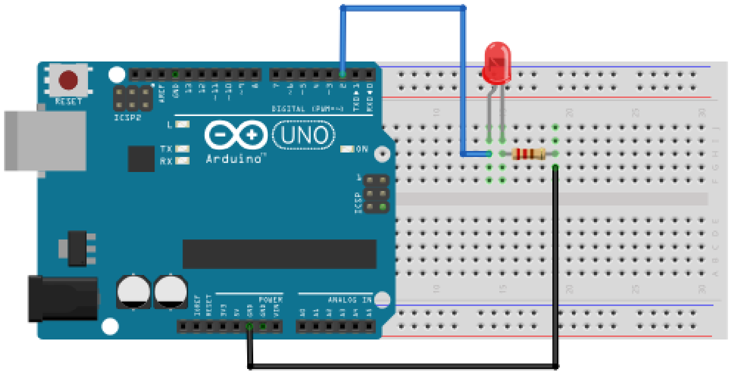


🟥 Normally used for positive supply.

🟦 Normally used for negative or ground supply.

▬ Normally used for connecting components - Components placed in the same row will be connected.

1.6 DIGITAL OUTPUT – LED



```
// -------------------- LED BLINK --------------------- //
// LED connected to digital pin 2
const int LED_PIN = 2;

//the setup function runs once when you press reset,
//power the board or open serial monitor
void setup() {
  // initialize LED_PIN as an output
  pinMode (LED_PIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite (LED_PIN, HIGH); //turn the LED on (HIGH)
  delay (1000);                 //wait for a second
  digitalWrite (LED_PIN, LOW);  //turn the LED off (LOW)
  delay (1000);                 //wait for a second
}
```

1.7 millis()

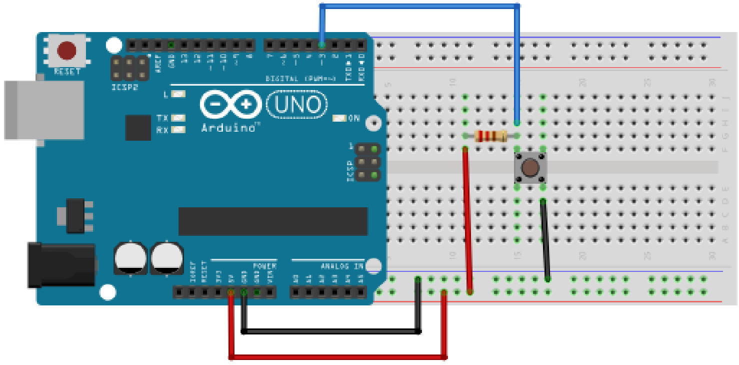
```
// ------------------- millis() ----------------------- //

unsigned long currentMillis;
void setup (){
  //open Serial port
  Serial.begin (9600);
}

void loop (){
  //store current time
  currentMillis = millis();
  // print current time
  Serial.println (currentMillis);
}
```

1.8 REPLACING delay() WITH millis()

```
// -------------- LED BLINK W/O DELAY ------------------ //
const int LED_PIN = 2;

unsigned long previousMillis;
int ledState;

void setup (){
  // initialize LED_PIN as an OUTPUT
  pinMode (LED_PIN, OUTPUT);
}

void loop (){
  if (millis() - previousMillis > 1000){
    ledState = !ledState;
    digitalWrite (LED_PIN, ledState);
    // store the last time you blink the LED
    previousMillis = millis();
  }
}
```

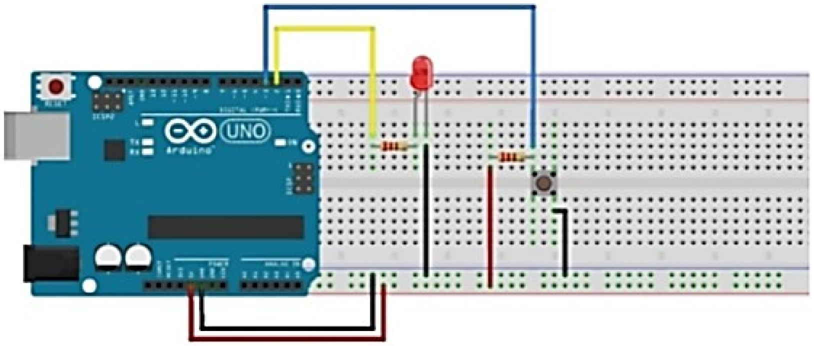
MODULE 2

2.1 DIGITAL INPUT – PUSH BUTTON



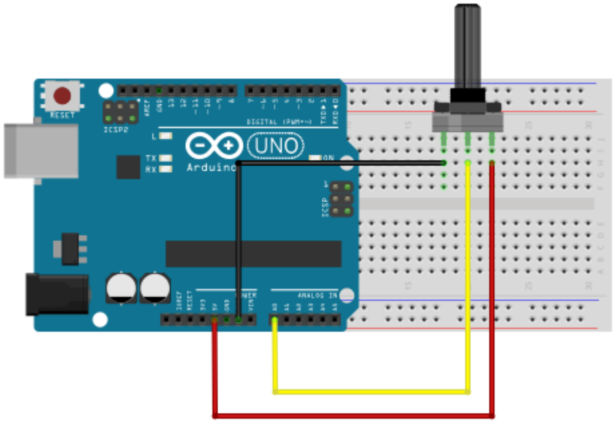
```
// ------------------- PUSH BUTTON -------------------- //
// push button connected to digital pin 3
const int BUTTON_PIN = 3;

void setup() {
  Serial.begin (9600);
  // initialize BUTTON_PIN as input
  pinMode (BUTTON_PIN, INPUT);
}

void loop() {
  // read  button state either HIGH or LOW (1 or 0)
  int buttonState = digitalRead (BUTTON_PIN);

  // print button state
  Serial.println (buttonState);
}
```

2.2 DIGITAL INPUT & OUTPUT



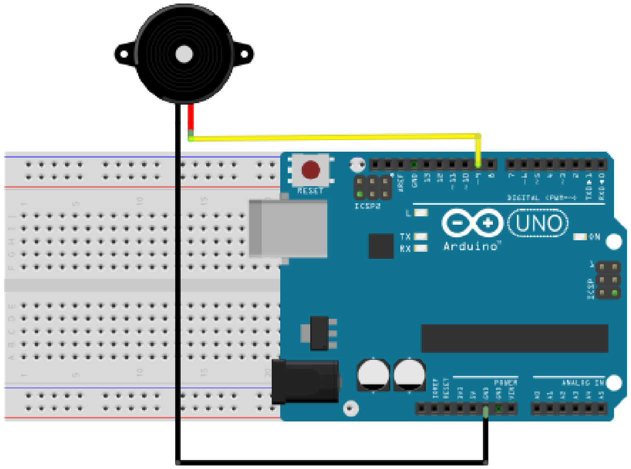
```
// ---------------- LED & PUSH BUTTON ----------------- //
// LED connected to digital pin 2
const int LED_PIN = 2;
// push button connected to digital pin 3
const int BUTTON_PIN = 3;

void setup() {
  // initialize LED_PIN as output
  pinMode (LED_PIN, OUTPUT);
  // initialize BUTTON_PIN as input
  pinMode (BUTTON_PIN, INPUT);
}

void loop() {
  // read  button state either HIGH or LOW (1 or 0)
  int buttonState = digitalRead (BUTTON_PIN);

  if (!buttonState)
    digitalWrite (LED_PIN, HIGH);
  else
    digitalWrite (LED_PIN, LOW);
}
```

2.3 ANALOG INPUT - POTENTIOMETER



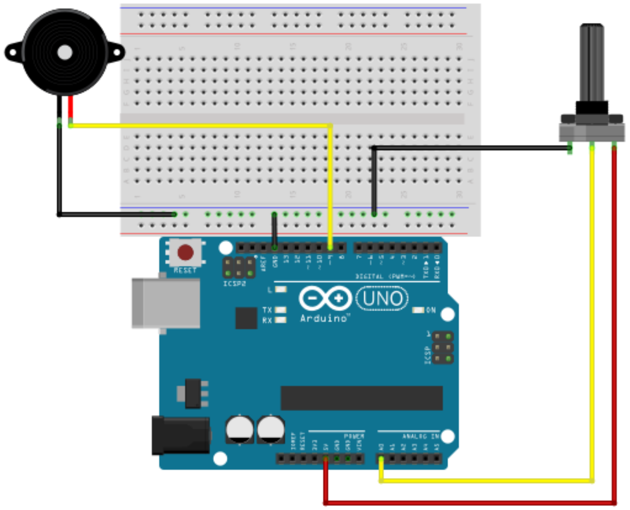
```
// ------------------- POTENTIOMETER -------------------- //
// potentiometer connected to analog pin A0
const int POT_PIN = A0;

void setup() {
  Serial.begin (9600);
  // initialize POT_PIN as input
  pinMode (POT_PIN, INPUT);
}

void loop() {
  // read analog value from potentiometer
  int potVal = analogRead (POT_PIN);

  // print the input value
  Serial.println (potVal);
}
```

2.4 ANALOG OUTPUT - BUZZER



```
// ---------------------- BUZZER ---------------------- //
// buzzer connected to digital pwm pin 9
const int BUZZER_PIN = 9;
// maximum pwm value
const int MAX_PWM = 255;
// to hold the value of pwm. By default, initialize to 0
int pwmVal;

void setup() {
  // initialize BUZZER_PIN as output
  pinMode (BUZZER_PIN, OUTPUT);
}

void loop() {
  // write analog value to buzzer (0 - 255)
  analogWrite (BUZZER_PIN, pwmVal);
  pwmVal++; // val increase by 1

  if (pwmVal > MAX_PWM) // if pwmVal greater than 255
    pwmVal = 0;         // initialize back to 0
  delay (50);      // delay so that we can see the changes
}
```

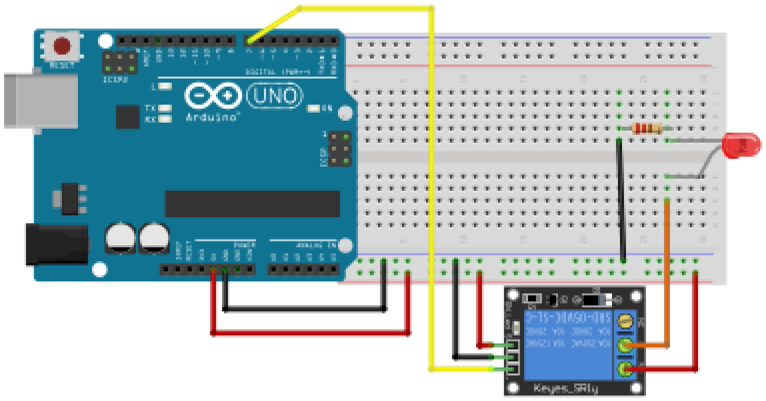
2.5 ANALOG INPUT & OUTPUT



```
// ------ USING POTENTIOMETER TO CONTROL BUZZER ------- //
// buzzer connected to digital pwm pin 9
const int BUZZER_PIN = 9;
// potentiometer connected to analog pin A0
const int POT_PIN = A0;

void setup() {
  // initialize BUZZER_PIN as output
  pinMode (BUZZER_PIN, OUTPUT);
  // initialize POT_PIN as input
  pinMode (POT_PIN, INPUT);
}

void loop() {
  // read potentiometer analog value
  int potVal = analogRead (POT_PIN);
  // scale it to use with buzzer
  potVal = map (potVal, 0, 1023, 0, 255);
  // write value from potentiometer to buzzer
  analogWrite (BUZZER_PIN, potVal);
}
```

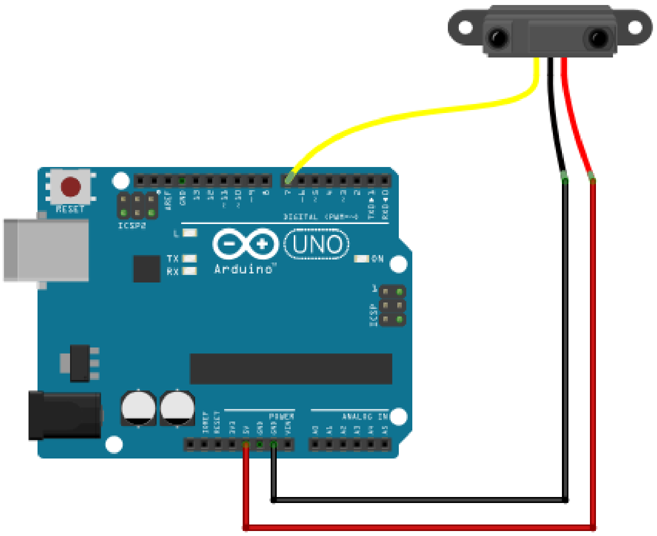
MODULE 3

3.1 RELAY SWITCH (DIGITAL OUTPUT)



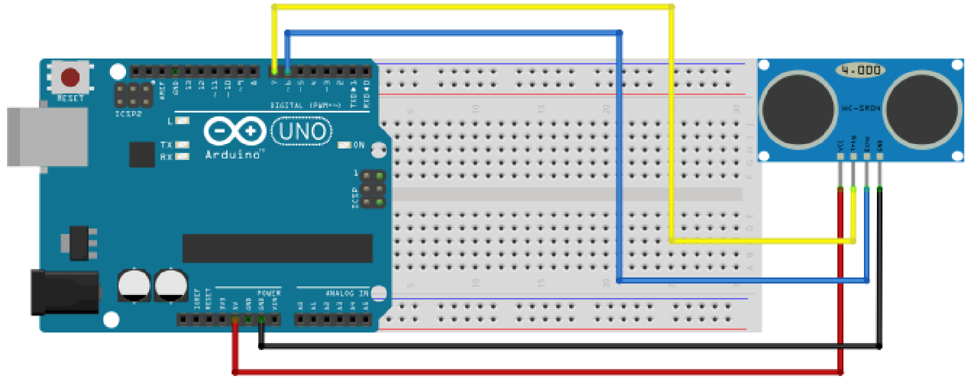
```
// ---------------------- RELAY ----------------------- //
// relay connected to digital pin 7
const int RELAY_PIN = 7;

void setup() {
  // initialize  RELAY_PIN as output
  pinMode (RELAY_PIN, OUTPUT);
}

void loop() {
  // toggle the pin state each 2s
  digitalWrite (RELAY_PIN, LOW);
  delay (2000);
  digitalWrite (RELAY_PIN, HIGH);
  delay (2000);
}
```

3.2 IR SENSOR (DIGITAL INPUT)



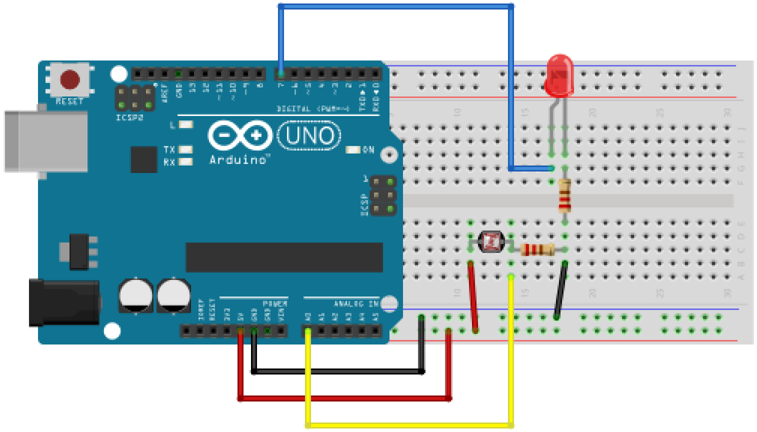
```
// -------------------- IR SENSOR --------------------- //
// LED pin on digital pin 13
const int LED_PIN = 13;
// IR pin connected to digital pin 7
const int IR_PIN = 7;

void setup() {
  // initialize LED_PIN as output
  pinMode (LED_PIN, OUTPUT);
  // initialize IR_PIN as input
  pinMode (IR_PIN, INPUT);
}

void loop() {
  // read IR sensor state (HIGH or LOW)
  int irState = digitalRead (IR_PIN);
  // write the reverse of the IR state to LED
  digitalWrite (LED_PIN, !irState);
}
```

3.3 ULTRASONIC SENSOR



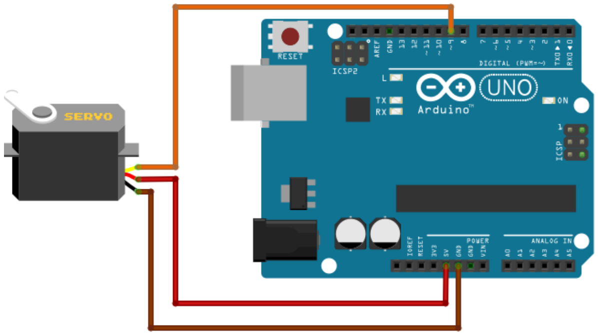
```
// ---------------- ULTRASONIC SENSOR ----------------- //
const int ECHO_PIN = 9;
const int TRIG_PIN = 10;
const float SOUND_SPEED = 0.034; //unit : cm/µs

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

void loop() {
  digitalWrite (TRIG_PIN, LOW);
  delayMicroseconds (2);
  digitalWrite (TRIG_PIN, HIGH);
  delayMicroseconds (10);
  digitalWrite (TRIG_PIN, LOW);

  long duration = pulseIn (ECHO_PIN, HIGH);
  int distance = (duration*SOUND_SPEED)/2;

  Serial.print ("Distance : ");
  Serial.println (distance);
}
```

3.4 LDR



```
// ----------------------- LDR ------------------------ //
// led connected to digital pin 7
const int LED_PIN = 7;
// ldr connected to analog pin A0
const int LDR_PIN = A0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(LDR_PIN, INPUT);
}

void loop() {
  int ldrVal = analogRead(LDR_PIN);
  if (ldrVal <= 100)
    digitalWrite(LED_PIN, HIGH);
  else
    digitalWrite(LED_PIN, LOW);
}
```

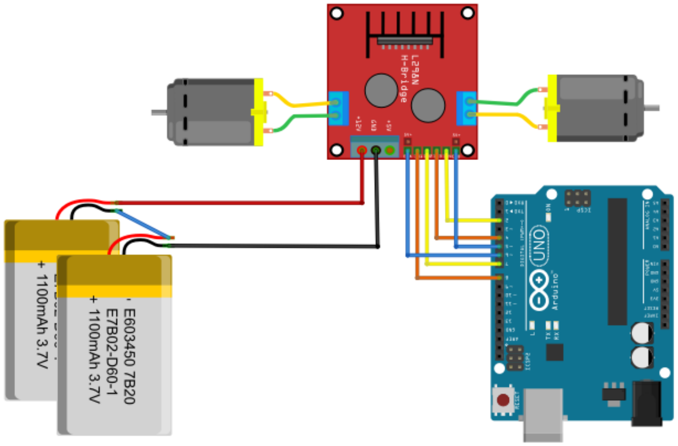
3.5 SERVO



```
// --------------------- SERVO ------------------------ //
#include <Servo.h>

const int SERVO_PIN = 9;
Servo myservo;  // create servo object to control a servo

const int MIN_ANGLE = 0, MAX_ANGLE = 180;

void setup() {
  myservo.attach(SERVO_PIN);  // attaches the servo pin
}

void loop() {
  static int pos;

  for (pos = MIN_ANGLE; pos <= MAX_ANGLE; pos++) {
    // tell servo to go to position in variable 'pos'
    myservo.write(pos);
    // waits 15ms for the servo to reach the position
    delay(15);
  }

  for (pos = MAX_ANGLE; pos >= MIN_ANGLE; pos--) {
    myservo.write(pos);
    delay(15);
  }
}
```

MODULE 4

4.1 MOTOR



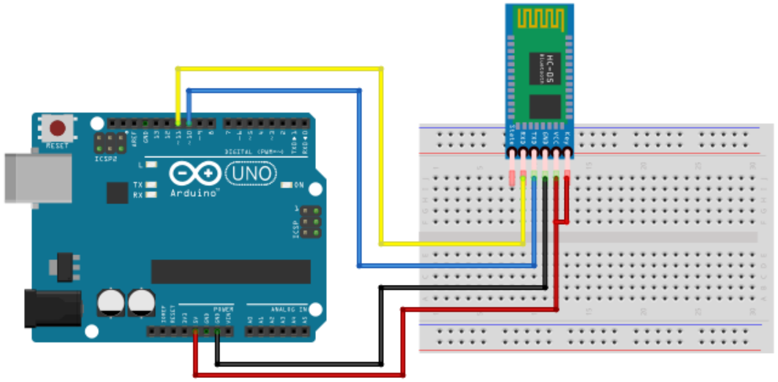
```
// ----------- CONTROLLING MOTOR WITH L298N ------------ //
const int EN_A_LEFT  = 6;
const int IN_1_LEFT  = 8;
const int IN_2_LEFT  = 7;
const int EN_B_RIGHT = 5;
const int IN_3_RIGHT = 4;
const int IN_4_RIGHT = 2;

void forward (int pwmLeft, int pwmRight){
  digitalWrite (IN_1_LEFT, HIGH);
  digitalWrite (IN_2_LEFT, LOW);
  analogWrite (EN_A_LEFT, pwmLeft);
  digitalWrite (IN_3_RIGHT, HIGH);
  digitalWrite (IN_4_RIGHT, LOW);
  analogWrite (EN_B_RIGHT, pwmRight);
}
```

```
void backward (int pwmLeft, int pwmRight){
  digitalWrite (IN_1_LEFT, LOW);
  digitalWrite (IN_2_LEFT, HIGH);
  analogWrite (EN_A_LEFT, pwmLeft);
  digitalWrite (IN_3_RIGHT, LOW);
  digitalWrite (IN_4_RIGHT, HIGH);
  analogWrite (EN_B_RIGHT, pwmRight);
}

void left (int pwmLeft, int pwmRight){
  digitalWrite (IN_1_LEFT, LOW);
  digitalWrite (IN_2_LEFT, HIGH);
  analogWrite (EN_A_LEFT, pwmLeft);
  digitalWrite (IN_3_RIGHT, HIGH);
  digitalWrite (IN_4_RIGHT, LOW);
  analogWrite (EN_B_RIGHT, pwmRight);
}

void right (int pwmLeft, int pwmRight){
  digitalWrite (IN_1_LEFT, HIGH);
  digitalWrite (IN_2_LEFT, LOW);
  analogWrite (EN_A_LEFT, pwmLeft);
  digitalWrite (IN_3_RIGHT, LOW);
  digitalWrite (IN_4_RIGHT, HIGH);
  analogWrite (EN_B_RIGHT, pwmRight);
}
void stopp (){
  digitalWrite (IN_1_LEFT, LOW);
  digitalWrite (IN_2_LEFT, LOW);
  analogWrite (EN_A_LEFT, 0);
  digitalWrite (IN_3_RIGHT, HIGH);
  digitalWrite (IN_4_RIGHT, HIGH);
  analogWrite (EN_B_RIGHT, 0);
}
```

```
void setup() {
  pinMode (EN_A_LEFT, OUTPUT);
  pinMode (IN_1_LEFT, OUTPUT);
  pinMode (IN_2_LEFT, OUTPUT);
  pinMode (EN_B_RIGHT, OUTPUT);
  pinMode (IN_3_RIGHT, OUTPUT);
  pinMode (IN_4_RIGHT, OUTPUT);
}

void loop() {
  forward (255,255);
  delay (1000);
  forward (150,150);
  delay (1000);
  backward (150,150);
  delay (2000);
  left (150,150);
  delay (2000);
  right (150,150);
  delay (2000);
  stopp ();
  delay (2000);
}
```

MODULE 5

5.1 BLUETOOTH AT COMMAND



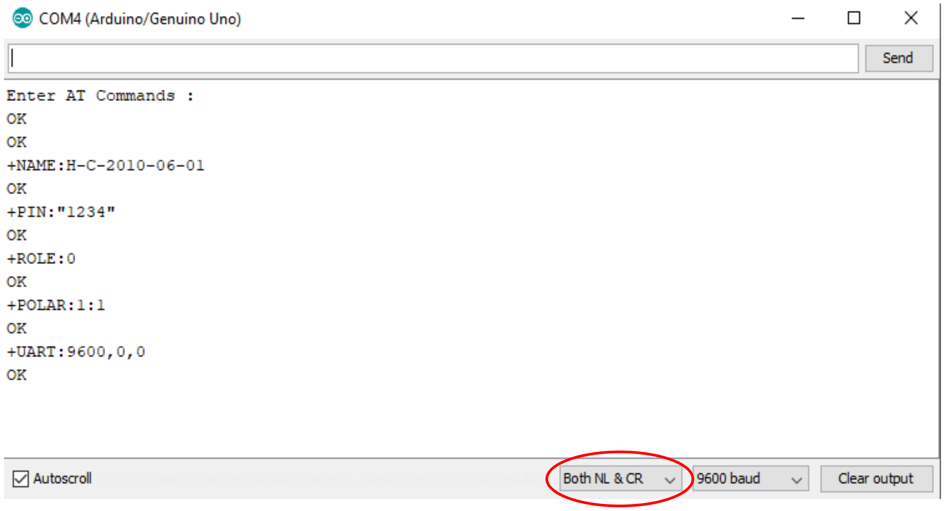
```
// ---------------- AT COMMAND MODE ------------------- //
#include <SoftwareSerial.h>

const int RX_PIN = 10;
const int TX_PIN = 11;

SoftwareSerial btSerial (RX_PIN, TX_PIN);

void setup() {
  Serial.begin (9600);
  btSerial.begin (38400); //BT default speed in AT command
  Serial.println ("Enter AT command : ");
}

void loop() {
  // keep reading from Serial Mon. and send data to HC-05
  if (Serial.available())
    btSerial.write (Serial.read());
  // keep reading from HC-05 and send data to Serial Mon.
  if (btSerial.available())
    Serial.write (btSerial.read());
}
```

Press and hold the on-board button on the Bluetooth module before applying power to it. You should notice the LED on-board the Bluetooth module has a long blink.

Type in these AT commands :

- AT (test command)
- AT+ORGL (restore default state)
- AT+NAME (set/check module name)
- AT+PSWD (set/check PIN code)
- AT+ROLE (set/check module mode)
- AT+POLAR (set/check LED I/O)
- AT+UART (set/check serial parameter)

Results (default state) :

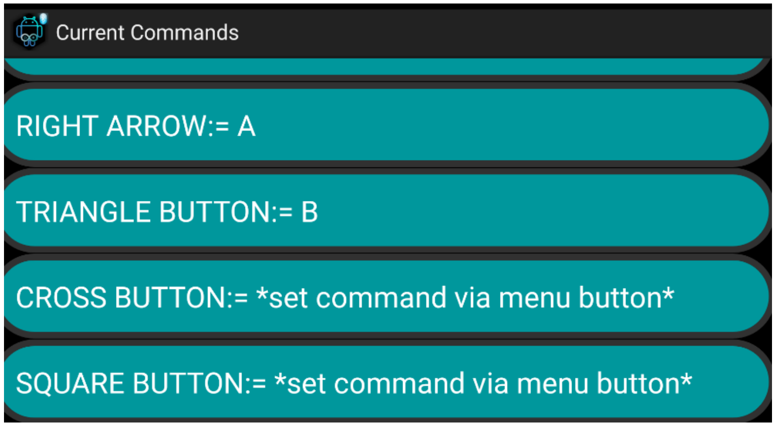


COM4 (Arduino/Genuino Uno)

```
Enter AT Commands :
OK
OK
+NAME:H-C-2010-06-01
OK
+PIN:"1234"
OK
+ROLE:0
OK
+POLAR:1:1
OK
+UART:9600,0,0
OK
```

☑ Autoscroll [✓ Both NL & CR] 9600 baud

5.2 ArduinoRC



Current Commands

RIGHT ARROW:= A

TRIANGLE BUTTON:= B

CROSS BUTTON:= *set command via menu button*

SQUARE BUTTON:= *set command via menu button*

```
// ------------------- ArduinoRC ---------------------- //
#include <SoftwareSerial.h>

const int RX_PIN = 11;
const int TX_PIN = 10;

const int LED_PIN =13;

SoftwareSerial btSerial (RX_PIN, TX_PIN);

void setup() {
  Serial.begin (9600);
  btSerial.begin (9600); //AT+UART baud rate
  pinMode (LED_PIN, OUTPUT);
}

void loop() {
  if (btSerial.available()){
    char input = btSerial.read();
    if (input == 'A' or input == 'B')//A = ARROW,B = BUTTON
      digitalWrite (LED_PIN, !digitalRead(LED_PIN));
  }
}
```

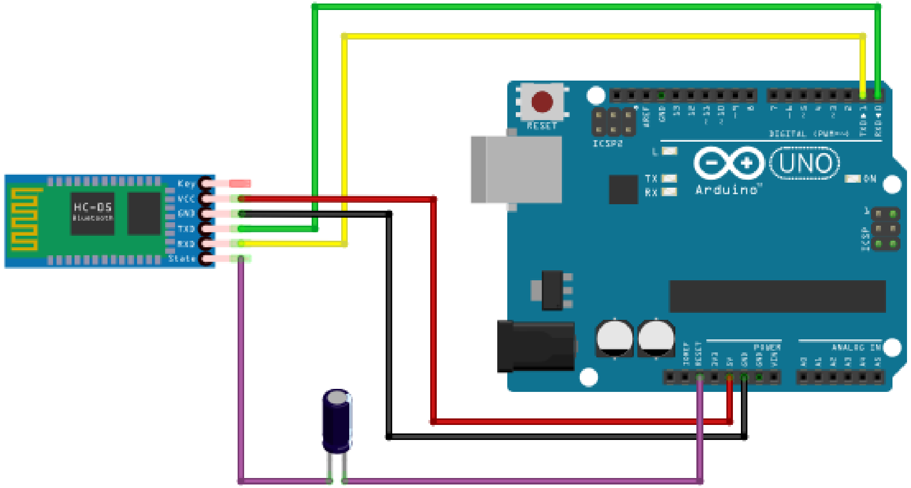
5.3 WIRELESS UPLOAD

i. Set up the Bluetooth AT command mode and enter the following commands in the serial terminal.

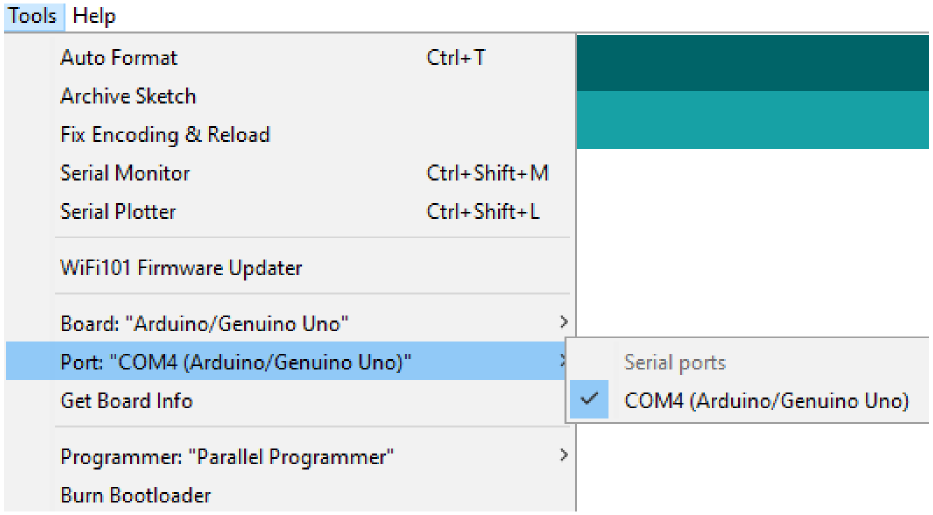
- AT+ORGL (optional, if you want to reset your Bluetooth)
- AT+ROLE = 0 (0 = Slave, 1=Master, 2=Slave-Loop)
- AT+POLAR=1,0
- AT+UART=115200,0,0 (baud, stop bit, parity)

ii. All commands shall get 'OK' as a return to indicate successful command.

iii. Disconnect the previous circuit and set up the circuit as below.



iv. Power up your Arduino and check the available COM port on Arduino IDE.



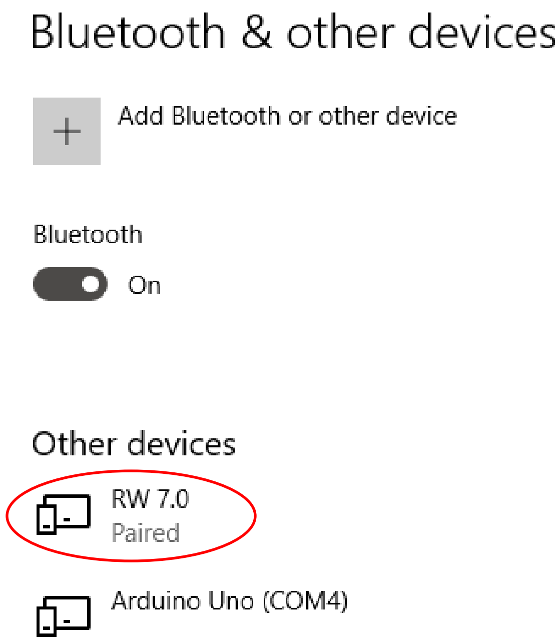
Board: "Arduino/Genuino Uno"

Port: "COM4 (Arduino/Genuino Uno)"

✓ COM4 (Arduino/Genuino Uno)

Programmer: "Parallel Programmer"

v. Open Bluetooth settings on your PC, scan for your Bluetooth device and connect/pair.



Bluetooth & other devices

+ Add Bluetooth or other device

Bluetooth

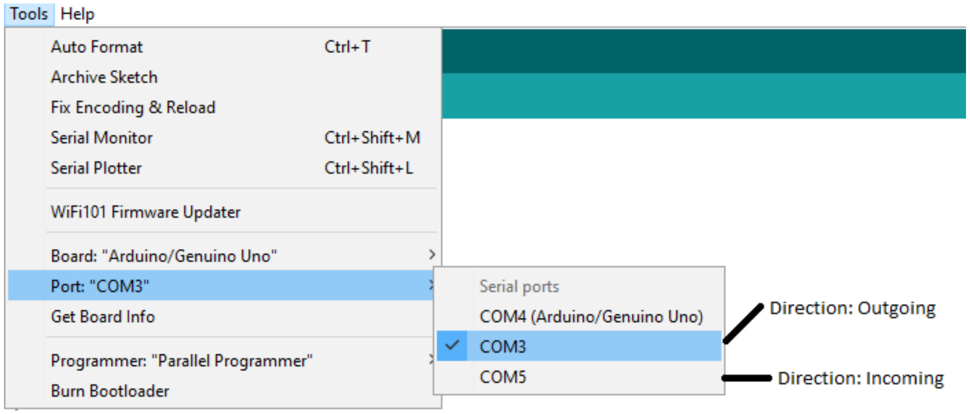
On

Other devices

RW 7.0
Paired

Arduino Uno (COM4)

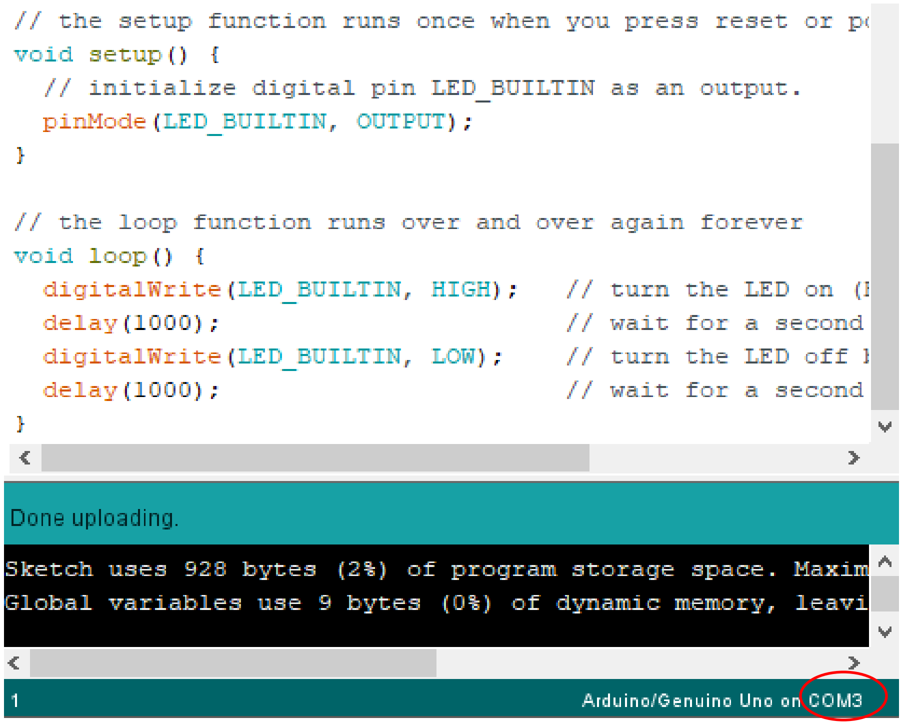
vi. Check the available COM port on the Arduino IDE again and you should notice two extra COM ports . Select the outgoing COM port.



Direction: Outgoing

Direction: Incoming

vii. Open a simple Blink code example from Arduino IDE and upload the code.



```
// the setup function runs once when you press reset or p
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off
  delay(1000);                       // wait for a second
}
```

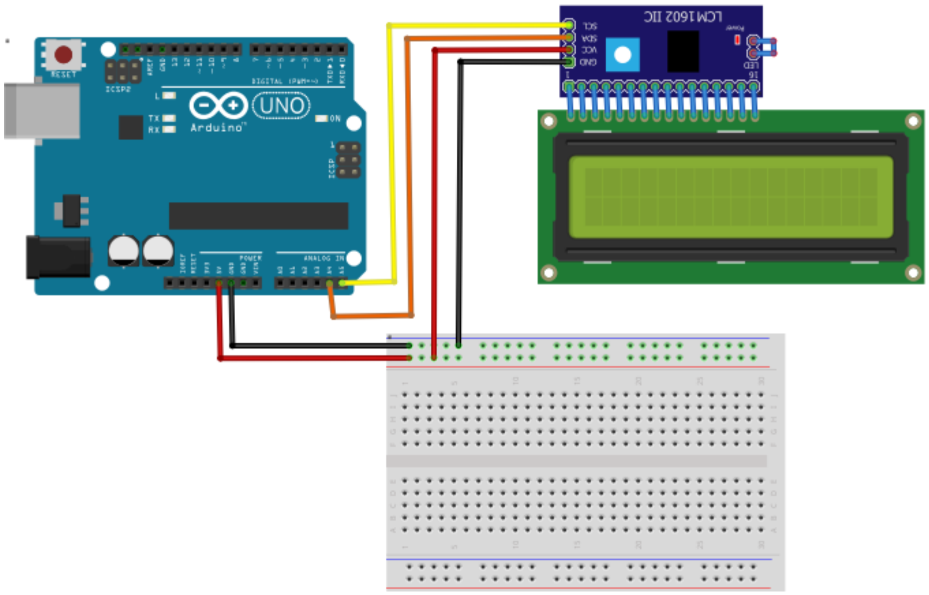
Done uploading.

```
Sketch uses 928 bytes (2%) of program storage space. Maxim
Global variables use 9 bytes (0%) of dynamic memory, leavi
```

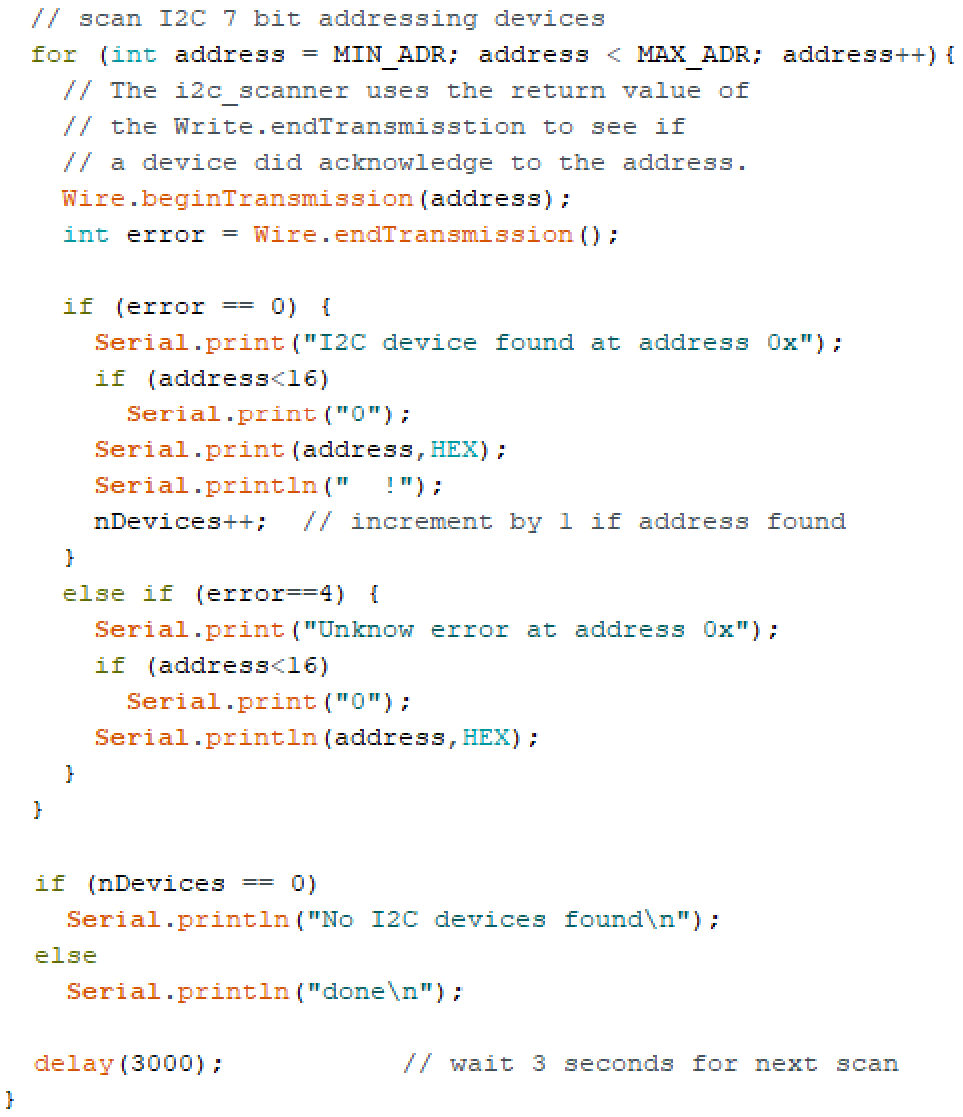
1 Arduino/Genuino Uno on COM3

MODULE 6

6.1 I2C Scanner

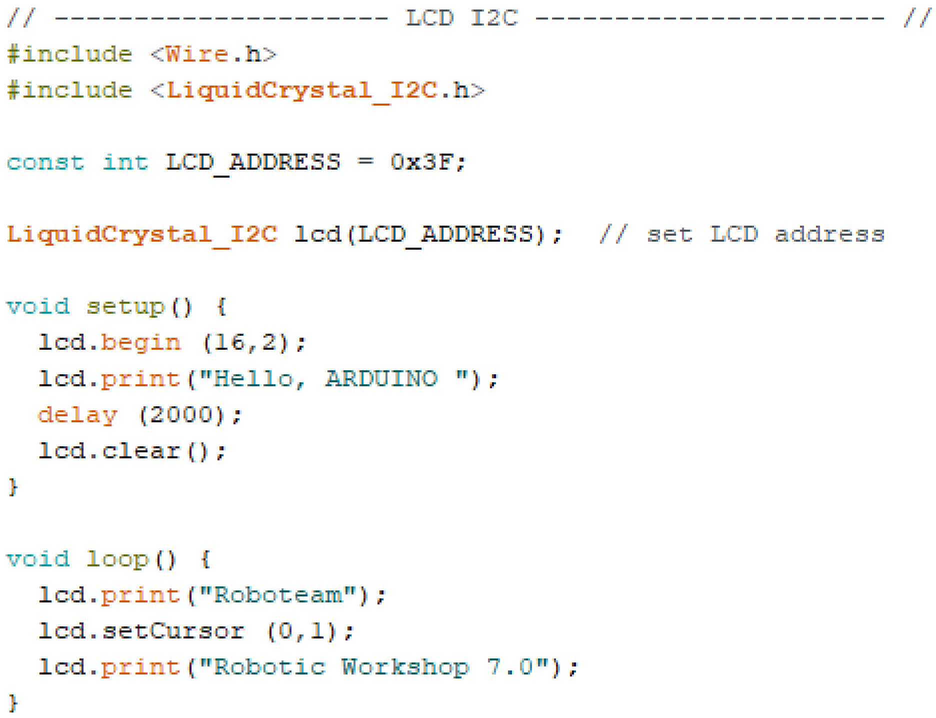


```
// -------------------- I2C Scanner--------------------- //
#include <Wire.h>

const int MIN_ADR = 0, MAX_ADR = 127;

void setup() {
  Wire.begin();
  Serial.begin (9600);
  Serial.println ("I2C Scanner\n");
}

void loop() {
  int nDevices; // number of address found

  Serial.println ("Scanning...");
```



```
  // scan I2C 7 bit addressing devices
  for (int address = MIN_ADR; address < MAX_ADR; address++){
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    int error = Wire.endTransmission();

    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");
      nDevices++;  // increment by 1 if address found
    }
    else if (error==4) {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }

  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(3000);           // wait 3 seconds for next scan
}
```

6.2 LCD I2C



```
// --------------------- LCD I2C ---------------------- //
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const int LCD_ADDRESS = 0x3F;

LiquidCrystal_I2C lcd(LCD_ADDRESS);  // set LCD address

void setup() {
  lcd.begin (16,2);
  lcd.print("Hello, ARDUINO ");
  delay (2000);
  lcd.clear();
}

void loop() {
  lcd.print("Roboteam");
  lcd.setCursor (0,1);
  lcd.print("Robotic Workshop 7.0");
}
```