UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

FACULTY OF ENGINEERING DESIGN AND TECHNOLOGY

COURSE: BACHELOR OF SCIENCE IN COMPUTER SCIENCE (BSCS)

COURSE UNIT: COMPUTER SCIENCE WORKSHOP

LECTURERS: MR. ISAAC NDAWULA & PROF. CHRISTOPHER SSENFUKA.

COMPUTER WORKSHOP REPORT

GROUP 2:

NAMES	**ACCESS NO.**	**REG.NO**
1. AROU ISAAC MAYOL	A94165	S21B23/012
2. AYEBARE MOSES	A95561	S21B23/032
3. MABIRA CONRAD	A94170	S21B23/017
4. MOGA MUZAMIL ABDUL WAHAB	A94166	S21B23/013

5. NKATA JOSHUA LUYOMBYA	A94161	S21B23/008
6. MUTUA BRIAN MUNG'OKA	A94174	S21B23/003
7. ALINDA MARTHA K	A98245	M22B23/001
8. MUKISA ISAIAH	A94160	S21B23/007
9. ATWIINE TIRZAH	A98238	M22B23/004
10. NAJJOBA TRACY MARJORIE	A95681	S21B23/034
11. KATUKUNDA ROCHELLE	A94169	S21B23/016
12. MUNJWOK JAMES JALALA	A98246	M22B23/007
13. NINZIZA RONIE	A95545	S21B23/031
14. DHELO DHEKANA	A95244	S21B23/024
15. AKAMPA IAN	A97756	M22B23/011

Contents

ARDUINO CODING

Arduino and arduino programming:

Assignment one: Arduino Coding

Objective:

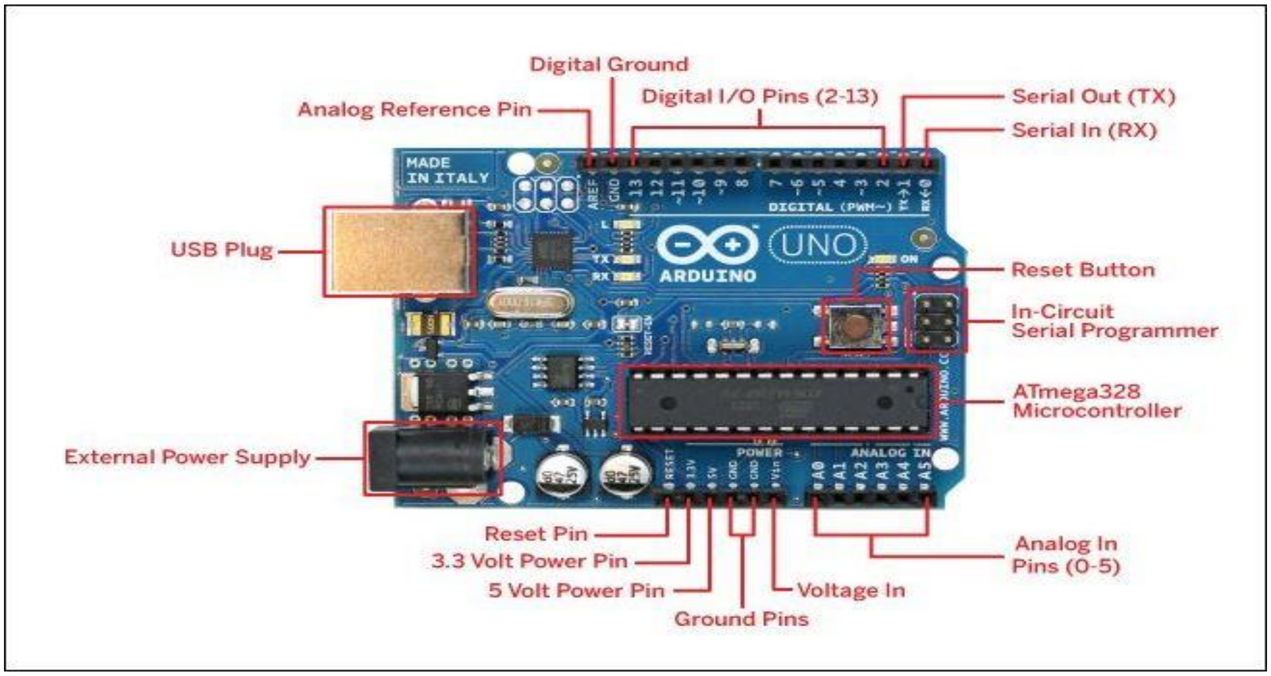
To introduce students to the Arduino code and software.

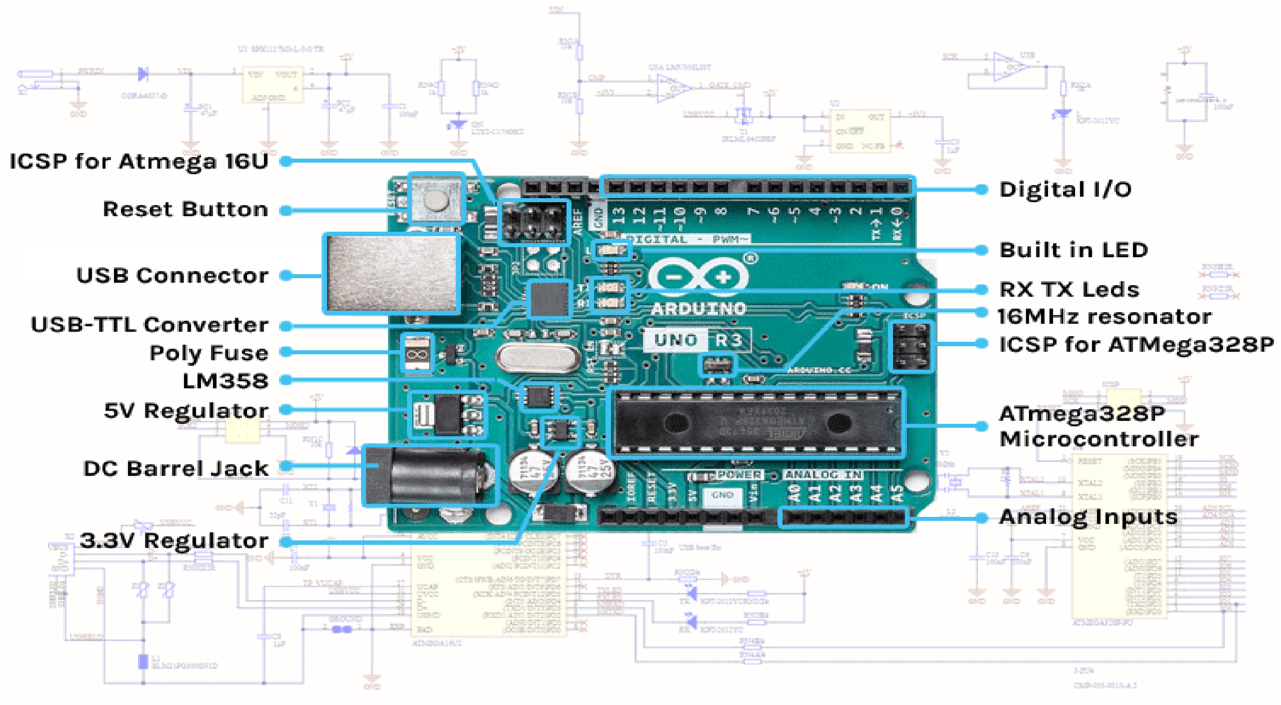
Arduino and Arduino programming:
Arduino is **an open-source electronics platform based on easy-to-use hardware and software**. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

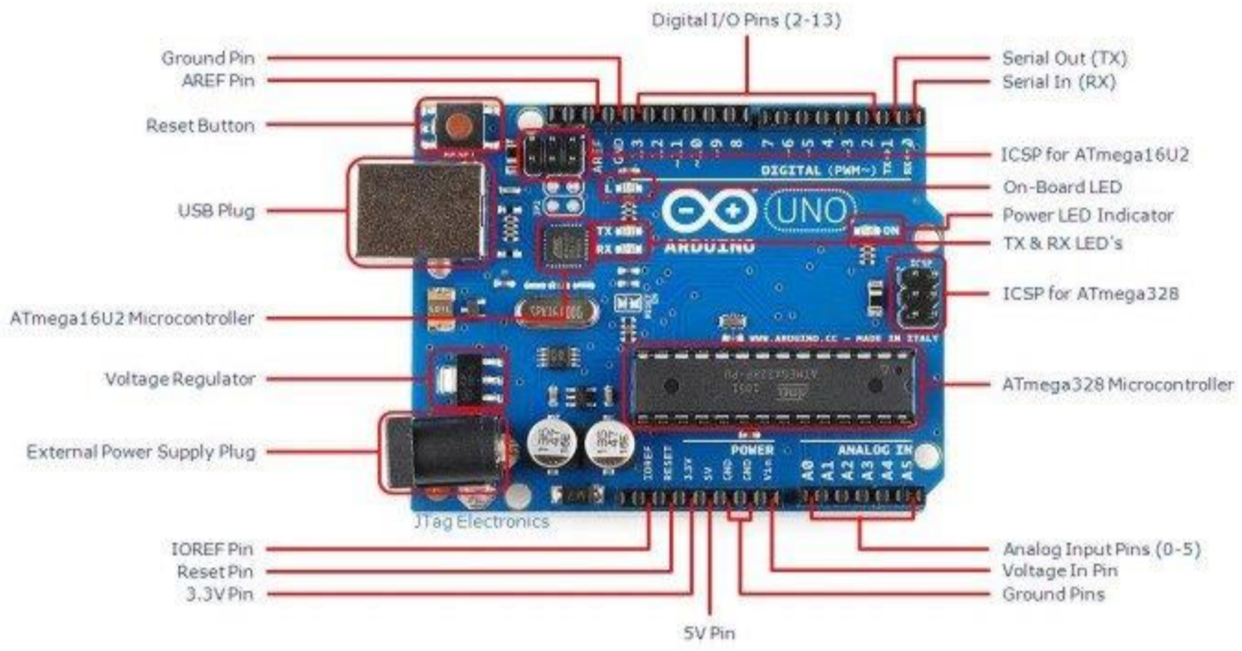
Types of Arduino: Arduino UNO, Red Board, LilyPad Arduino, Arduino Mega, Arduino Leonardo.

Activity

a) learn all the features of Arduino board.
b) Download and install Arduino Integrated Development Environment
c) Learn Arduino IDE software environment
d) Learn communication protocols in Arduino code
e) Learn about ATmega 328 Arduino
f) Learn about Arduino programming(typical structure of Arduino program(has two parts), syntax and knowledge about circuits(parallel and series connection)).


Digital Ground
Analog Reference Pin
Digital I/O Pins (2-13)
Serial Out (TX)
Serial In (RX)
USB Plug
Reset Button
In-Circuit Serial Programmer
ATmega328 Microcontroller
External Power Supply
Reset Pin
3.3 Volt Power Pin
5 Volt Power Pin
Ground Pins
Voltage In
Analog In Pins (0-5)


ICSP for Atmega 16U
Reset Button
USB Connector
USB-TTL Converter
Poly Fuse
LM358
5V Regulator
DC Barrel Jack
3.3V Regulator
Digital I/O
Built in LED
RX TX Leds
16MHz resonator
ICSP for ATMega328P
ATmega328P Microcontroller
Analog Inputs



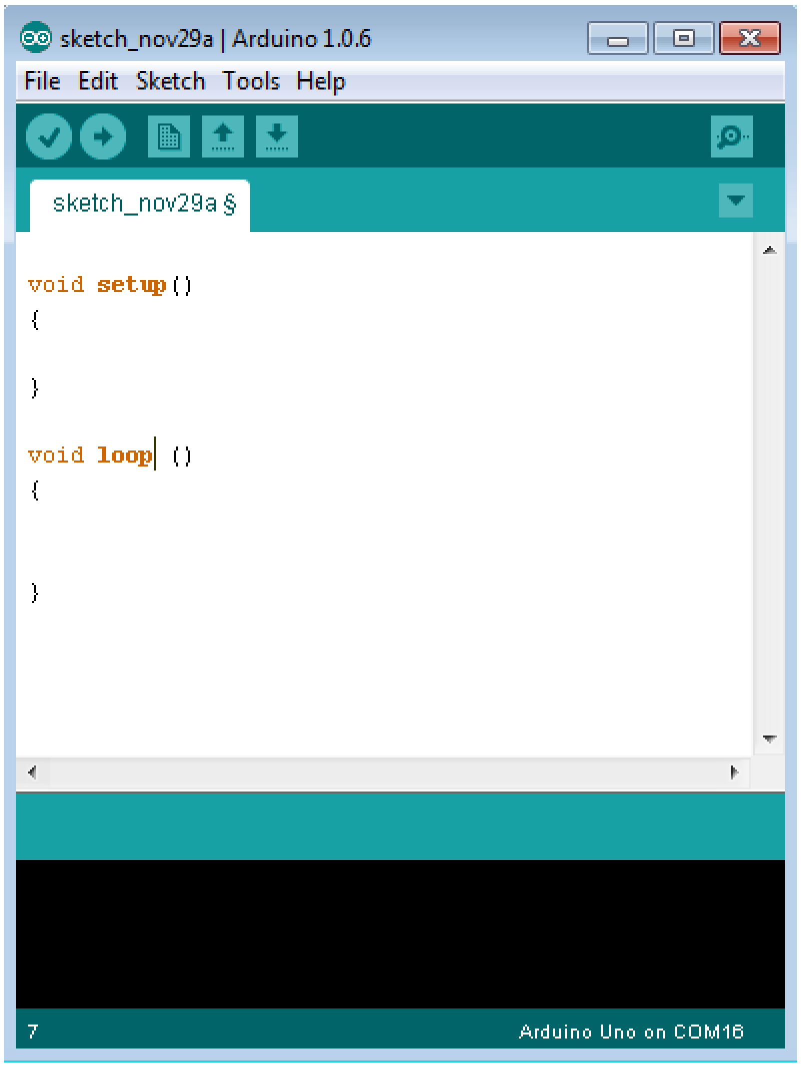
Software environment for the IDE

STRUCTURE

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consist of two main functions −

- Setup( ) function
- Loop( ) function



```
void setup()
{

}

void loop ()
{


}
```

Void setup ( ) {

}

- **PURPOSE** – The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.
- **INPUT** – -
- **OUTPUT** – -
- **RETURN** – -

Void Loop ( ) {

}

- **PURPOSE** – After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.
- **INPUT** – -

- **OUTPUT** – -
- **RETURN** – -

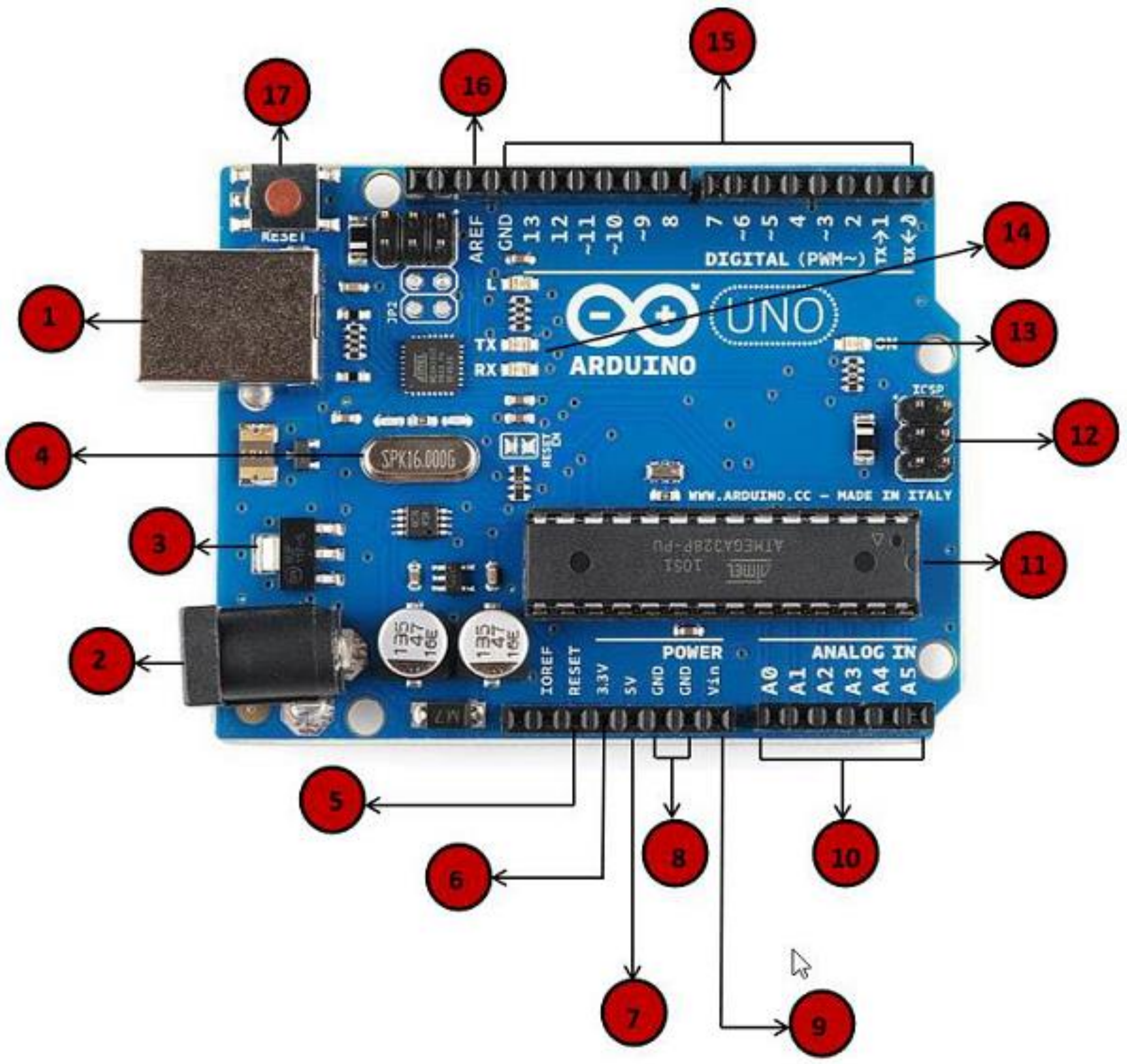
FUNCTIONS OF ARDUINO

Let's discuss some advantages of using functions in programming, which are listed below:

- It increases the readability of the code.
- It conceives and organizes the program.
- It reduces the chances of errors.
- It makes the program compact and small.
- It avoids the repetition of the set of statements or codes.
- It allows us to divide a complex code or program into a simpler one.
- The modification becomes easier with the help of functions in a program.

PRACTICAL APPLICATION

- Control of traffic lights
- Bluetooth interface for connectivity
- Used in temperature sensor
- Used in alarm system
- Used in weighing machines
- Used in parking lot counter



1	**Power USB** Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
3	**Voltage Regulator**

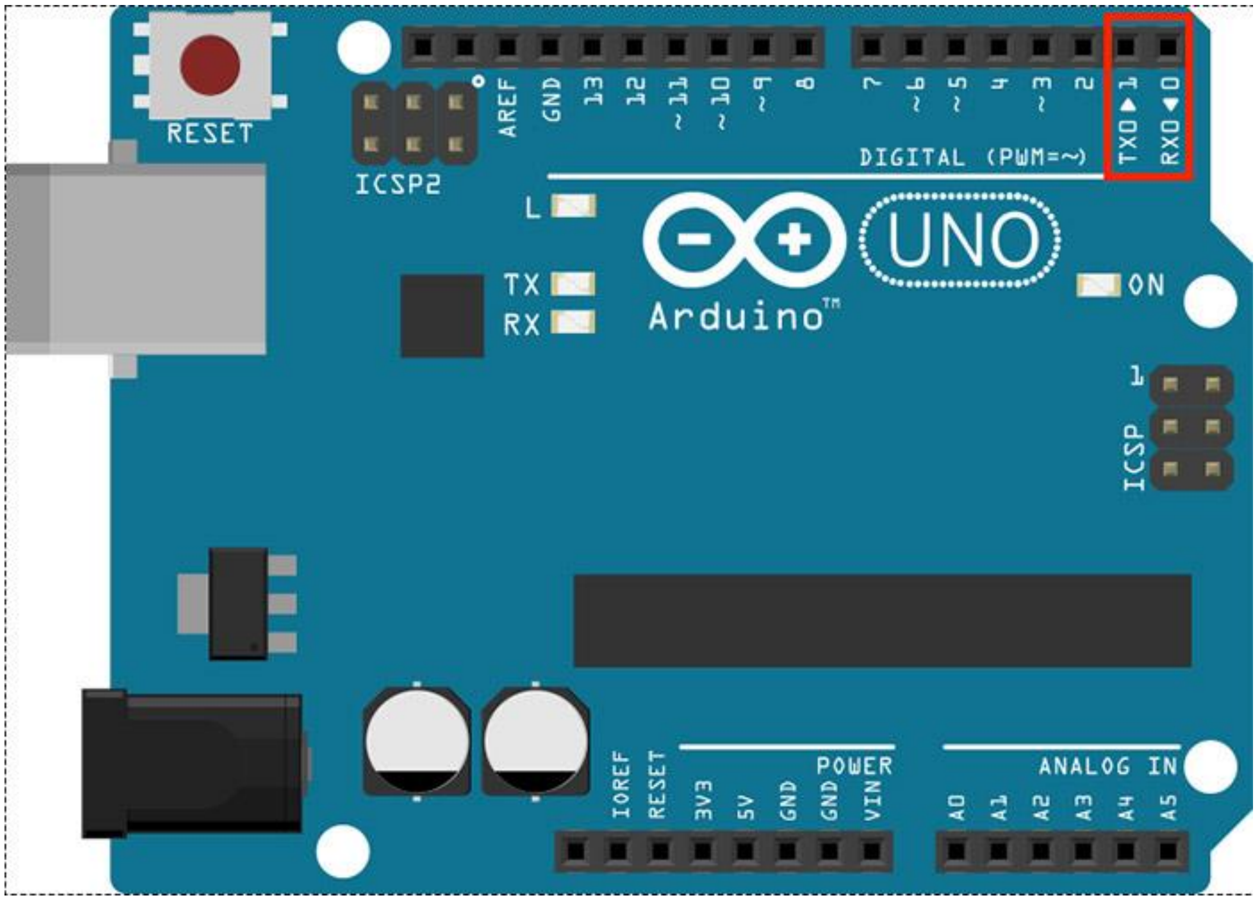
	The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
4	**Crystal Oscillator** The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
5,17	**Arduino Reset** You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).
6,7 8,9	**Pins (3.3, 5, GND, Vin)** • 3.3V (6) – Supply 3.3 output volt • 5V (7) – Supply 5 output volt • Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. • GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.
10	**Analog pins** The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.
11	**Main microcontroller** Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

12	**ICSP pin** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.
13	**Power LED indicator** This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.
14	**TX and RX LEDs** On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.
15	**Digital I/O** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.
16	**AREF** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

PROTOCOLS FOR DEVICE COMMUNICATION: UART, SPI and I^{2}C

1. UART Communication Protocol

UART is a form of *serial communication* because data is transmitted as sequential bits (we'll get to this in a bit). The wiring involved with setting up UART communication is very simple: one line for transmitting data (TX) and one line for receiving data (RX).

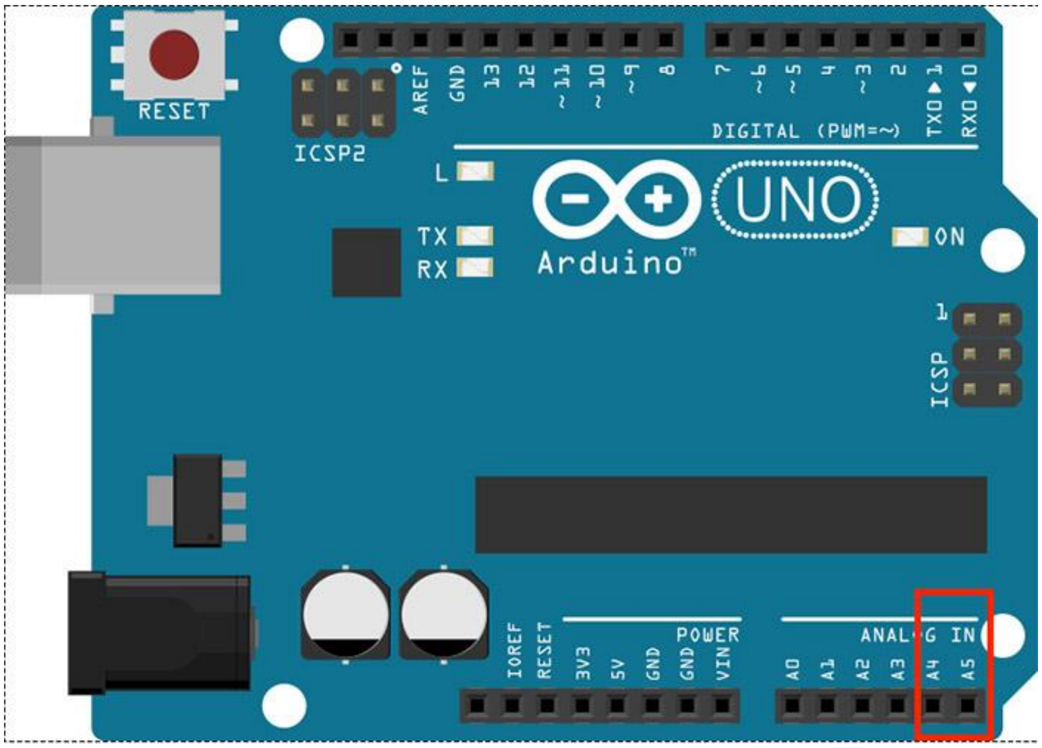


2. **SPI Communication Protocol**

3. **I^{2}C Communication Protocol**

Inter-integrated circuit (I^{2}C), pronounced either "i-squared-c" or "i-two-c," is the final communication protocol we'll cover in this tutorial. Though its implementation is the most complicated of the three protocols, I^{2}C addresses several drawbacks in the other communication protocols, giving it an advantage over the others in some applications. These include:

- The ability to connect multiple masters to multiple slaves
- Synchronicity (just like SPI), which means higher speed communication
- Simplicity: implementation only requires two wires and some resistors

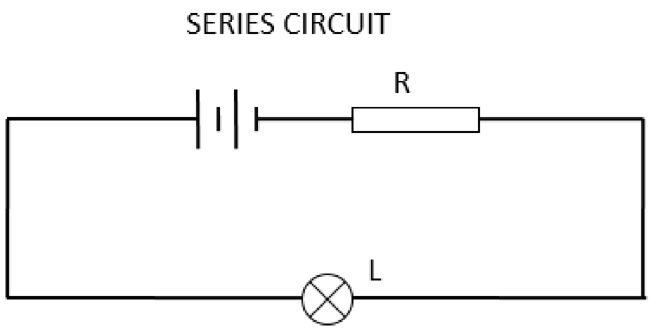


AT MEGA328P

ATmega328P is a high performance yet low power consumption 8-bit AVR microcontroller that's able to achieve the most single clock cycle execution of 131 powerful instructions thanks to its advanced RISC architecture. It can commonly be found as a processor in Arduino boards such as Arduino Fio and Arduino Uno

Arudino programming refers to **arduino IDE which is a special software running on your system that allows you to write sketches (synonym for program in Arduino language) for different Arduino boards**.

Arduino circuit boards offer you a collection of analog and digital I/O pins, Can connect different circuits and expansion boards. You can also use Arduino circuit boards for serial communication, burning code directly from your computer, and providing power.
And Syntax in Arduino signifies the rules need to be followed for the successful uploading of the Arduino program to the board. The syntax of Arduino is **similar to the grammar in English**. It means that the rules must be followed in order to compile and run our code successfully.
Parallel and series connections

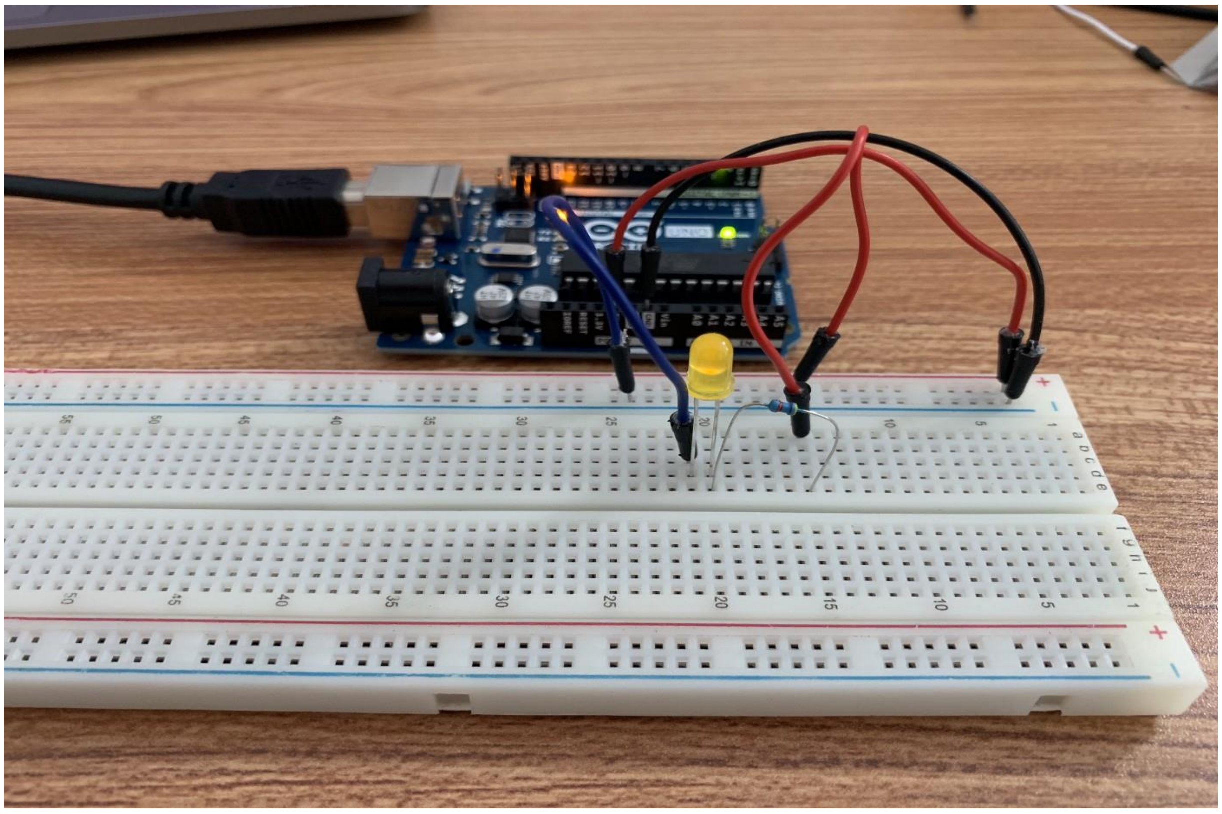


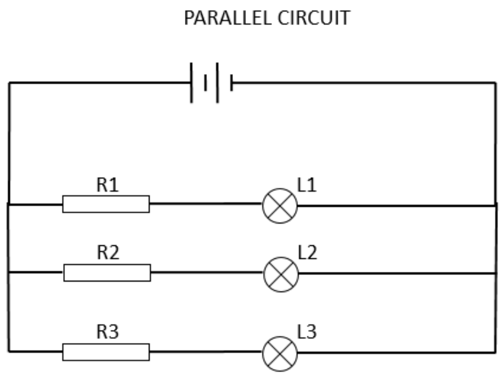
Current: The amount of current is the same through any component in a series circuit.

Resistance: The total resistance of any series circuit is equal to the sum of the individual resistances.

Voltage: The supply voltage in a series circuit is equal to the sum of the individual voltage drops.

AN IMAGE OF A SERIES CONNECTION



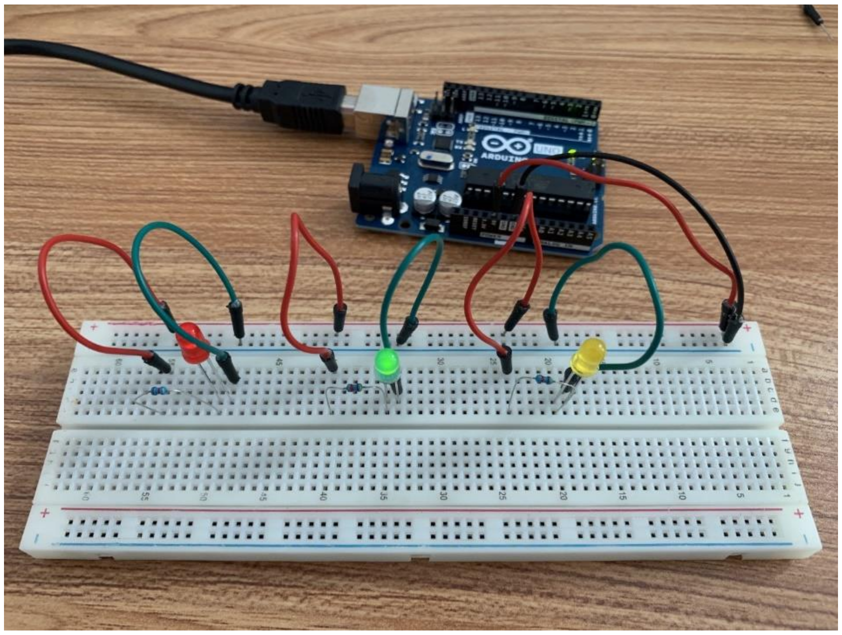


All components share the same voltage. Resistances diminish to equal a smaller, total resistance. Branch currents add to equal a larger, total current.

Voltage: Voltage is equal across all components in a parallel circuit.

Current: The total circuit current is equal to the sum of the individual branch currents.

Resistance: Individual resistances diminish to equal a smaller total resistance rather than add to make the total.



AN IMAGE OF A PARALLEL CIRCUIT

REPORT ON LED ACTIVITY

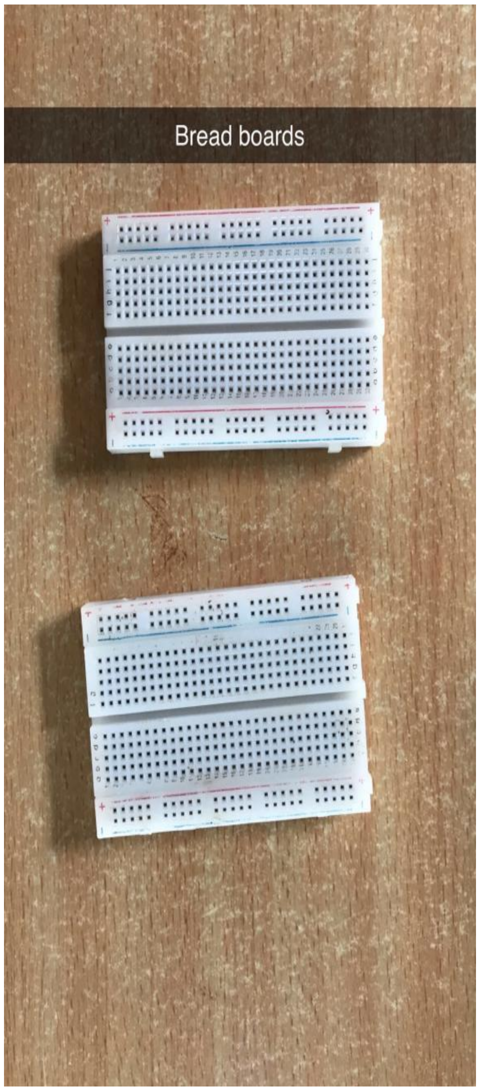
Assignment 2:

OBJECTIVE:

Controlling LED (Turning LEDS on and off, and then making them blink).

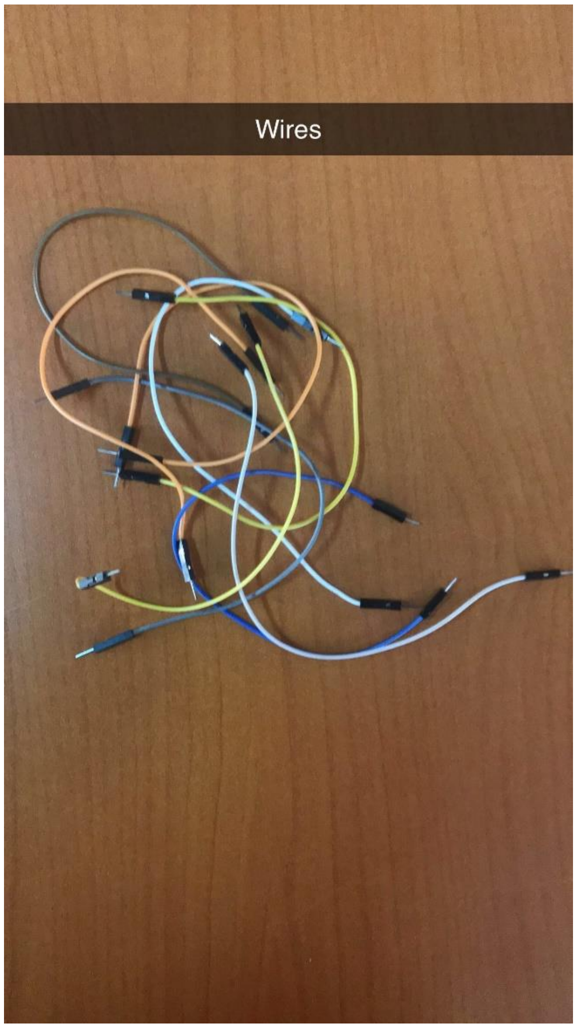
TOOLS:

A Breadboard, LED bulbs, wires, Arduino UNO R3, USB cable connecting Arduino to computer and computer.

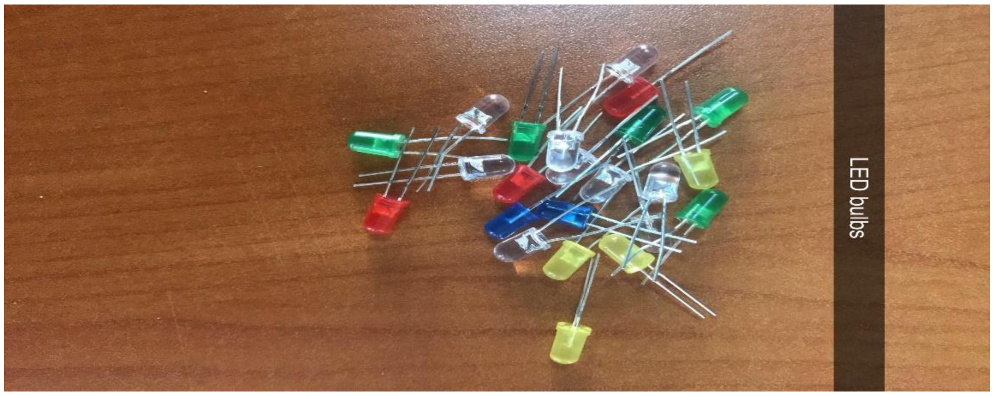


BREAD BOARDS

ARDUINO UNO R3



ARDUINO AND USB CABLE **WIRES**



LED BULBS

EXERCISE: Controlling LED lights.

The shorter leg(side) goes to the Ground and the longer leg(side) goes to the digital port.
We connected a blue bulb to digital port 4, connected a white bulb to digital port 9, and also used two bulbs(yellow and green) to digital port 13 getting current from the line and lastly connected bulbs(red and green) to digital port 13 getting current from the line.

PSEUDO CODE:

First write the code to turn the LED bulbs on, then write code to turn the bulbs off, and to make the bulbs blink after a given number of micro-seconds, write a code having both codes but include “delay();” in the loop function to make the bulbs turn on and off after the given time.

Turning all 6 LEDS on.

CODE

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(4,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  digitalWrite(9,HIGH);
  digitalWrite(4,HIGH);
}
```

Turning all LEDS off:

```
void setup() {
  // indicating the modes and ouput:
  pinMode(13,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(4,OUTPUT);
}

void loop() {
  // Turning the bulbs off :
  digitalWrite(13,LOW);
  digitalWrite(9,LOW);
  digitalWrite(4,LOW);
}
```

Making the LEDS blink:

```
Making it blink after 100 milliseconds
  void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(4,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  delay(100);
  digitalWrite(13,LOW);

  digitalWrite(9,HIGH);
  delay(100);
  digitalWrite(9,LOW);

  digitalWrite(4,HIGH);
  delay(100);
  digitalWrite(4,LOW);
}
```

USING THREE BULBS CONNECTING TO DIGITAL PORTS 7, 10 AND 13.

PSEUDO CODE:

First write the code to turn the LED bulbs on, then write code to turn the bulbs off, and to make the bulbs blink after a given number of micro-seconds, write a code having both codes but include “delay();” in the loop function to make the bulbs turn on and off after the given time.

Turning the three LEDS on:

CODE

```
void setup() {
  // indicating the modes and ouput:
  pinMode(7,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop() {
  // Turning the bulbs off :
  digitalWrite(7,HIGH);
  digitalWrite(10,HIGH);
  digitalWrite(13,HIGH);
}
```

Turning the three LEDS off:

CODE

```
void setup() {
  // indicating the modes and ouput:
  pinMode(7,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop() {
  // Turning the bulbs off :
  digitalWrite(7,LOW);
  digitalWrite(10,LOW);
  digitalWrite(13,LOW);
}
```

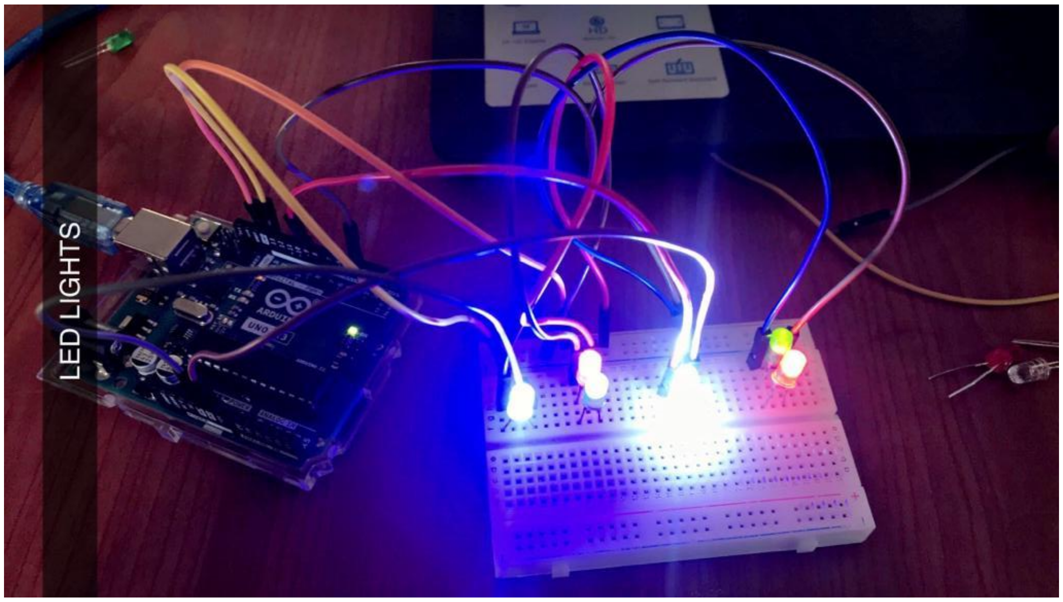
Making the LEDS blink:

CODE

```
void setup() {
  // indicating the modes and ouput:
  pinMode(7,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop() {
  // Turning the bulbs off :
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  digitalWrite(10,HIGH);
  delay(100);
  digitalWrite(10,LOW);

  digitalWrite(13,HIGH);
  delay(100);
  digitalWrite(13,LOW);
}
```



AN IMAGE OF LED LIGHTS TURNED ON USING THE ARDUINO

REPORT ON DC AND SEVOR MOTORS:

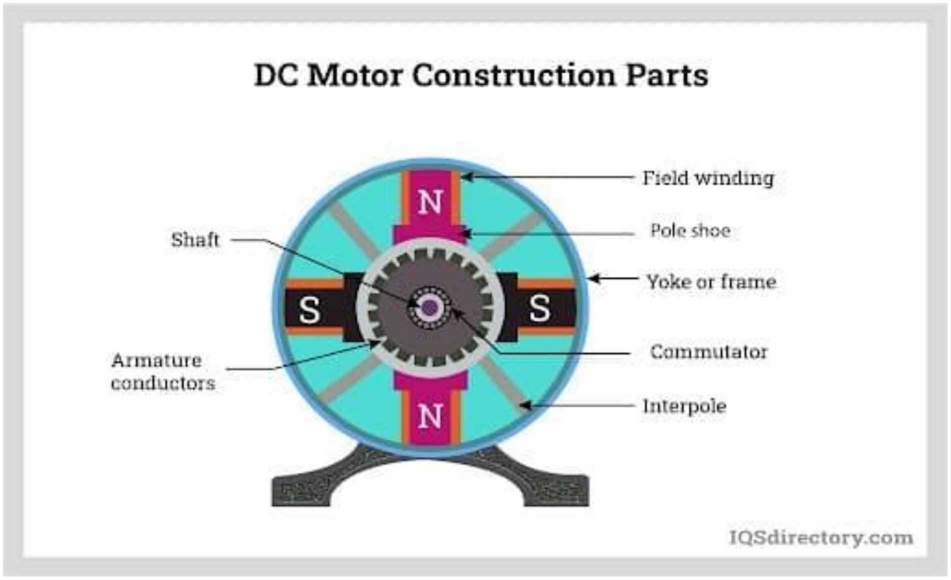
ASSIGNMENT 3: LINE TRACKING

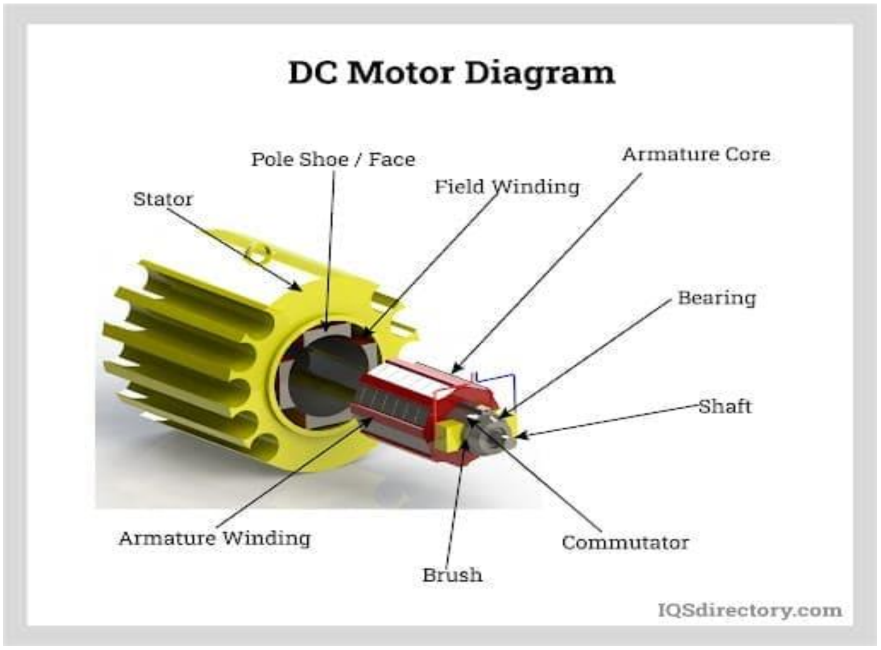
3. Using DC and Sevor motors:

3.1. What is a DC motor?
3.2. Types of DC motors
3.3. How the DC motor works
3.4. Uses/ Applications of DC motors
3.5. What is a sevor motor
3.6. Types of sevor motor
3.7. How sevor motor works
3.8. Uses/ Applications of sevor motor
3.9. Activity: using sevor motor

DC MOTOR:

A DC motor or direct current motor is an electrical machine that transforms electrical energy into mechanical energy by creating a magnetic field that is powered by direct current. When a DC motor is powered, a magnetic field is created in its stator. The field attracts and repels magnets on the rotor; this causes the rotor to rotate. To keep the rotor continually rotating, the commutator that is attached to brushes connected to the power source supply current to the motors wire windings.





Why DC motors are preferred compared to other types of motors:

DC motors have the ability to control their speed, which is a necessity for industrial machinery. DC motors are able to immediately start, stop, and reverse an essential factor for controlling the operation of production equipment.

TYPES OF DC MOTORS:

1. Permanent Magnet DC Motors.
2. Series DC Motors.
3. Shunt DC Motors.
4. Compound DC Motors.

HOW THE DC MOTOR WORKS:

A DC motor is based on the idea that when a current carrying conductor is placed in a magnetic field, it produces mechanical force. The direction of the force is determined by the left hand rule. Since DC motors and DC generators have the same construction, they can be used interchangeably.

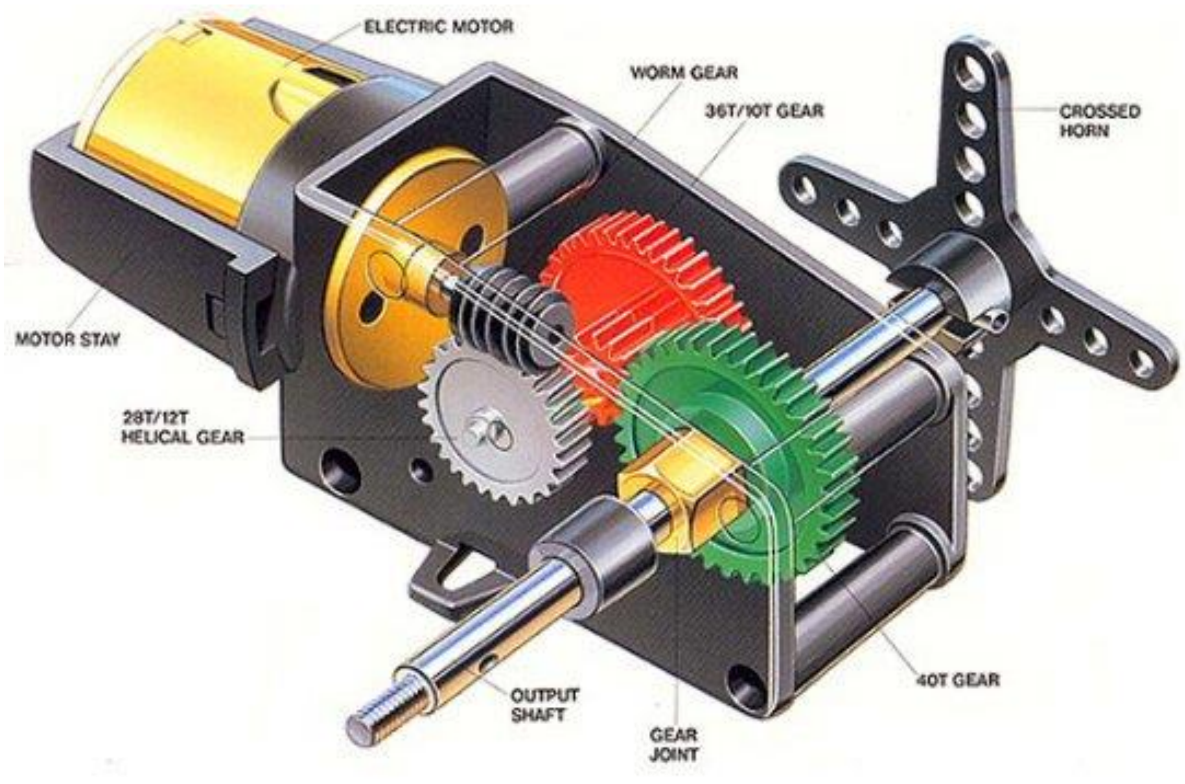
USES/ APPLICATIONS OF THE DC MOTOR:

DC motors are used in any number of applications since they have a high starting torque compared to induction motors. Applications of the DC motor include;

1. Diesel Electric Locomotives
2. Electric Vehicles
3. Cranes
4. Conveyor Systems
5. Ceiling Fans
6. Pump Drives
7. Elevators

SEVOR MOTORS:

A servo motor is a self-contained electrical device, that rotate parts of a machine with high efficiency and with great precision.



TYPES OF SEVOR MOTORS:

Types of Servo Motors are classified into different types based on their application.

1. AC servo motor and
2. DC servo motor.

HOW SEVOR MOTOR WORKS (CONTROLLING SERVO MOTOR):

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Servo motor works on **PWM (Pulse width modulation)** principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of **DC motor which is controlled by a variable resistor (potentiometer) and some gears**. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.

USES/ APPLICATIONS OF SEVOR MOTOR:

Servo Motor Applications are applied in many industrial and commercial systems and products such as with robotics where a servo motor is used at every “joint” of a robot to perform its precise angle of movement.

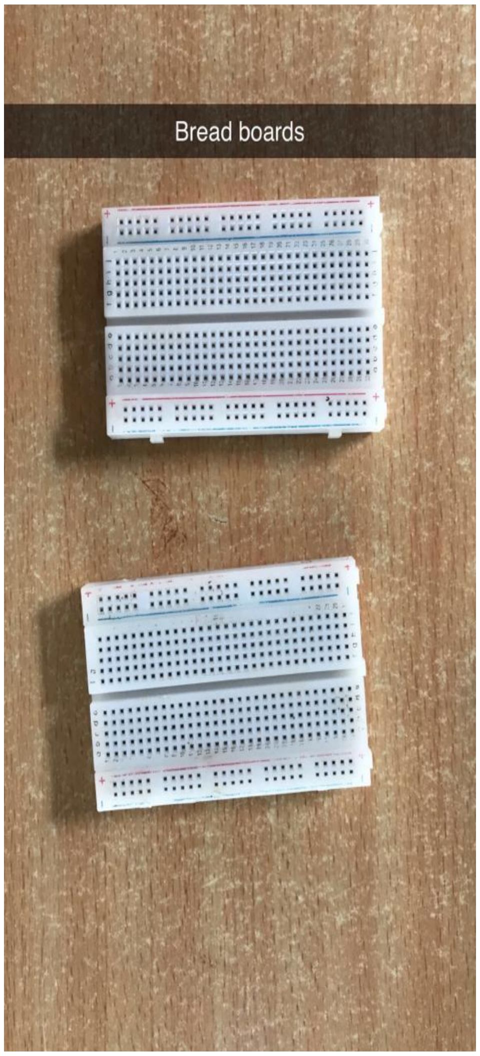
They are used for precise position and speed control at high torques in robotics.

Servo motors are used in computers, CD/DVD players, toys etc. Servos are extensively used in those application where a specific task is to be done repeatedly in a very precise manner.

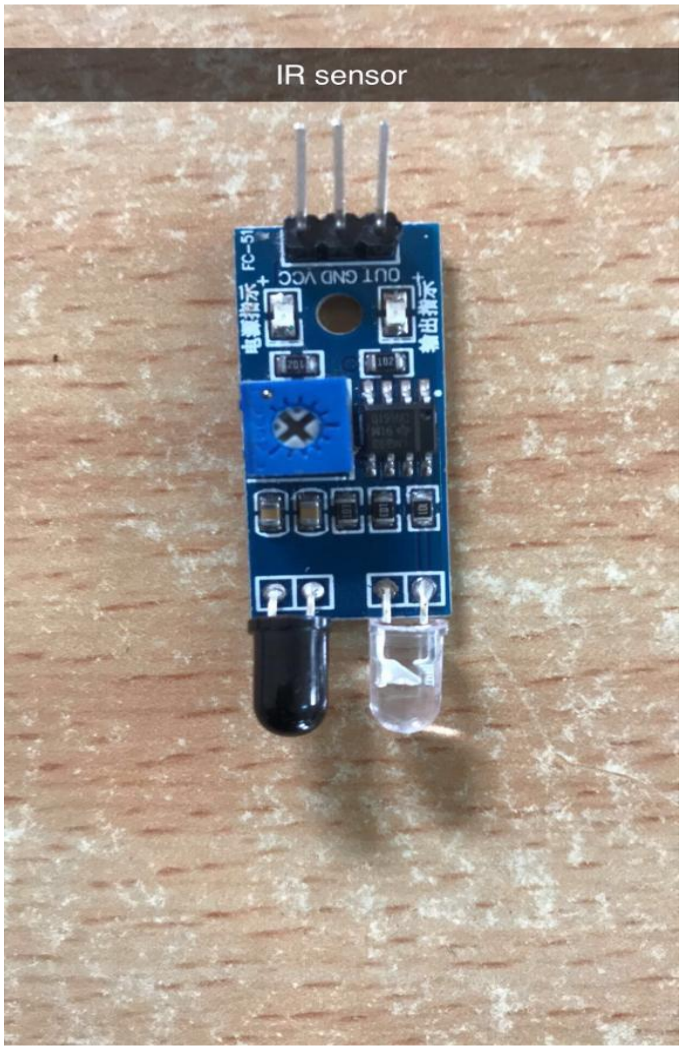
3.2 Activity: LINE TRACKING

Objective: To move the 2 wheel robot along the black line using the IR sensor to detect black line

Tools Used: Sevor motors, wires(male to male and female to male), Arduino UNO R3, IR sensor, Breadboards, and a USB cable.

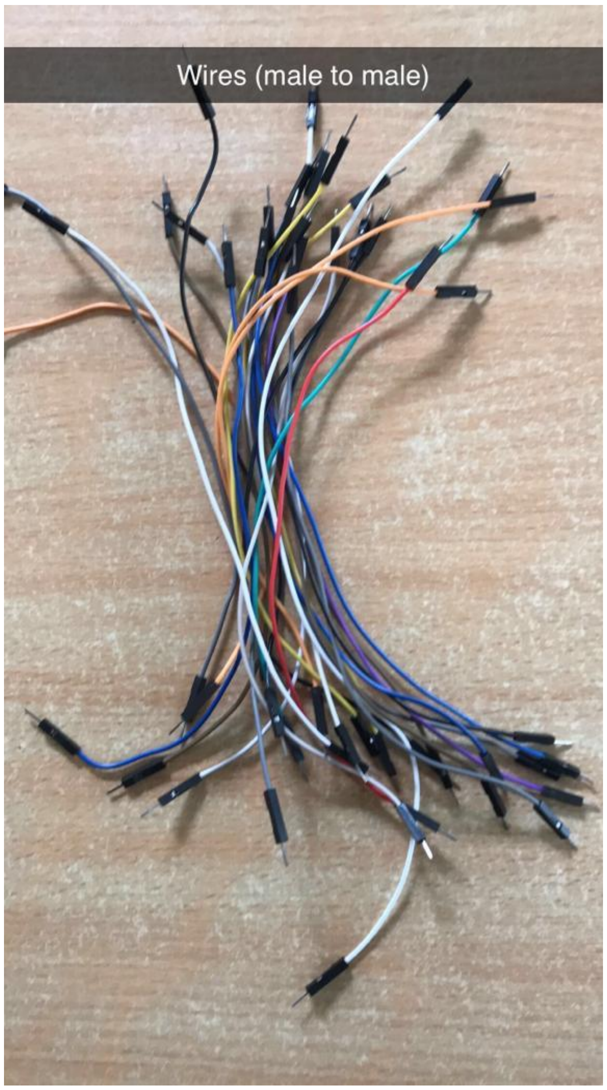


BREAD BOARDS

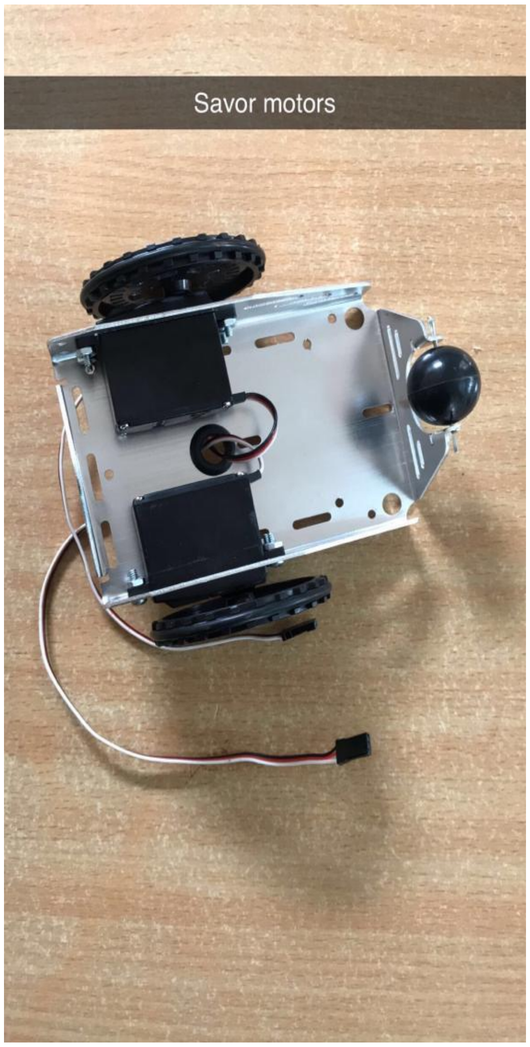


IR SENSOR

ARDUINO UNO R3



WIRES



SERVO MOTORS ARDUINO AND USB CABLE

CODE FOR LINE TRACKING ASSIGNMENT:

```
#include <Servo.h>
Servo myservo1,myservo2;
int sensor_pin=4;
int val;
void setup() {
  // put your setup code here, to run once:
 pinMode(sensor_pin,INPUT);


 myservo1.attach(9);
 myservo2.attach(10);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
 val=digitalRead(sensor_pin);
 if(val==0)
 {
  myservo1.write(0);
  myservo2.write(0);

}

 if(val==1)
 {
   myservo1.write(180);
   myservo2.write(0);
}
```

REPORT ON ASSIGNMENT 4 MAKING SEVOR MOTOR TURN 180 DEGRESS AND BACK:

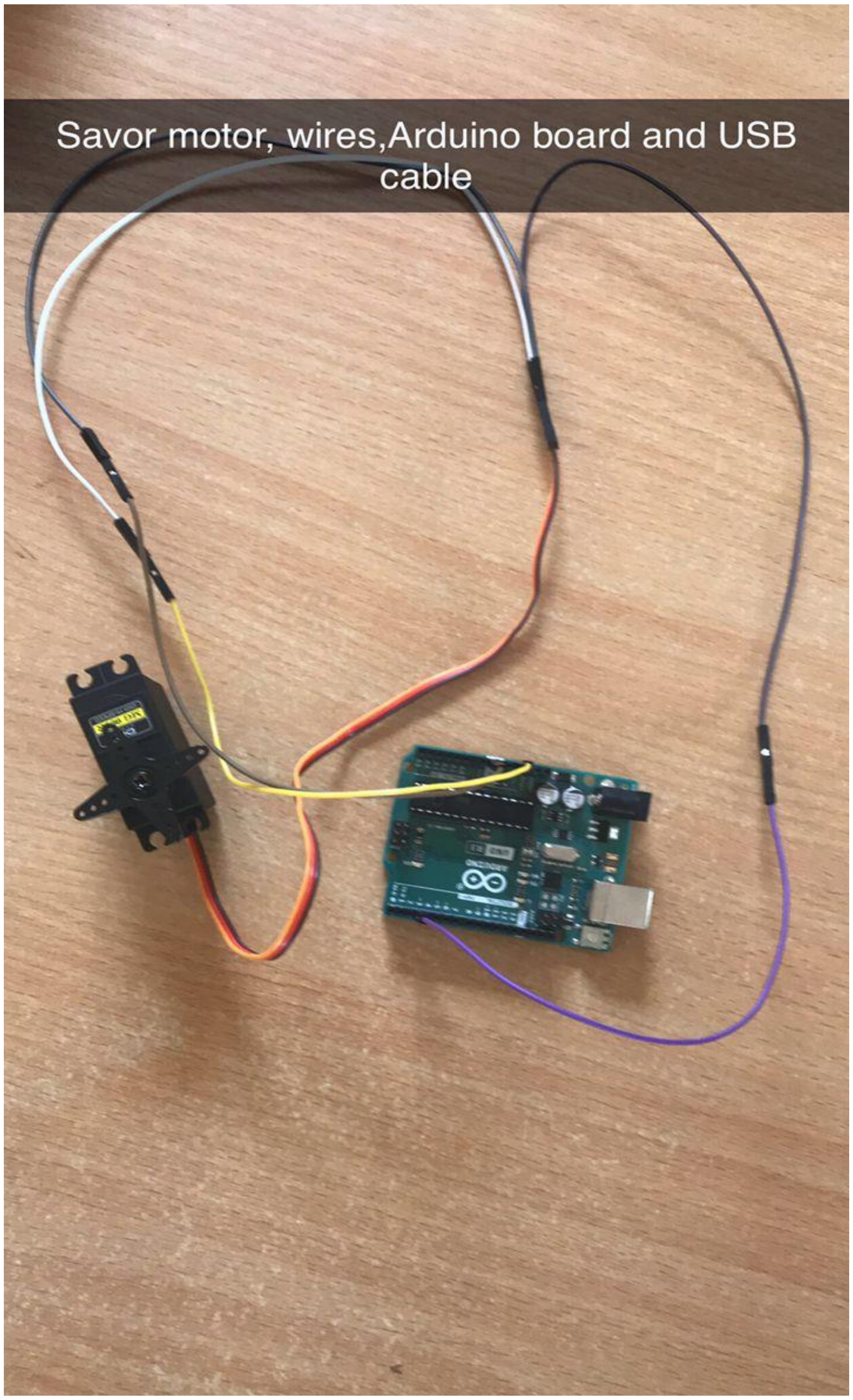
ASSIGNMENT 4:

OBJECTIVE:

To make a sevor motor turn 180 degrees and back

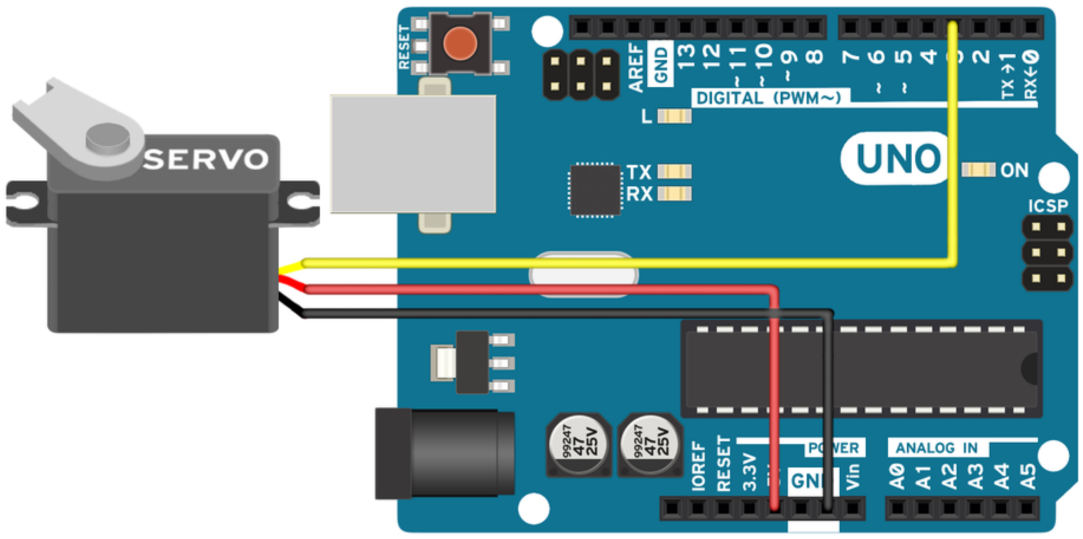
TOOLS:

sevor motor, wires, Arduino UNO R3 and USB Cable.



SERVO MOTOR, WIRES, ARDUINO AND A USB CABLE

CIRCUIT BOARD:



CODE

```
// Include the Servo library
#include <Servo.h>
// Declare the Servo pin
int servoPin = 3;
// Create a servo object
Servo Servo1;
void setup() {
   // We need to attach the servo to the used pin number
   Servo1.attach(servoPin);
}
void loop(){
   // Make servo go to 0 degrees
   Servo1.write(0);
   delay(1000);
   // Make servo go to 180 degrees
   Servo1.write(180);
   delay(1000);
}
```

REPORT ON 4WD OBSTACLE AVOIDANCE CAR (ROBOT):

Assignment 5: **CAR AVOIDING OBSTACLE**

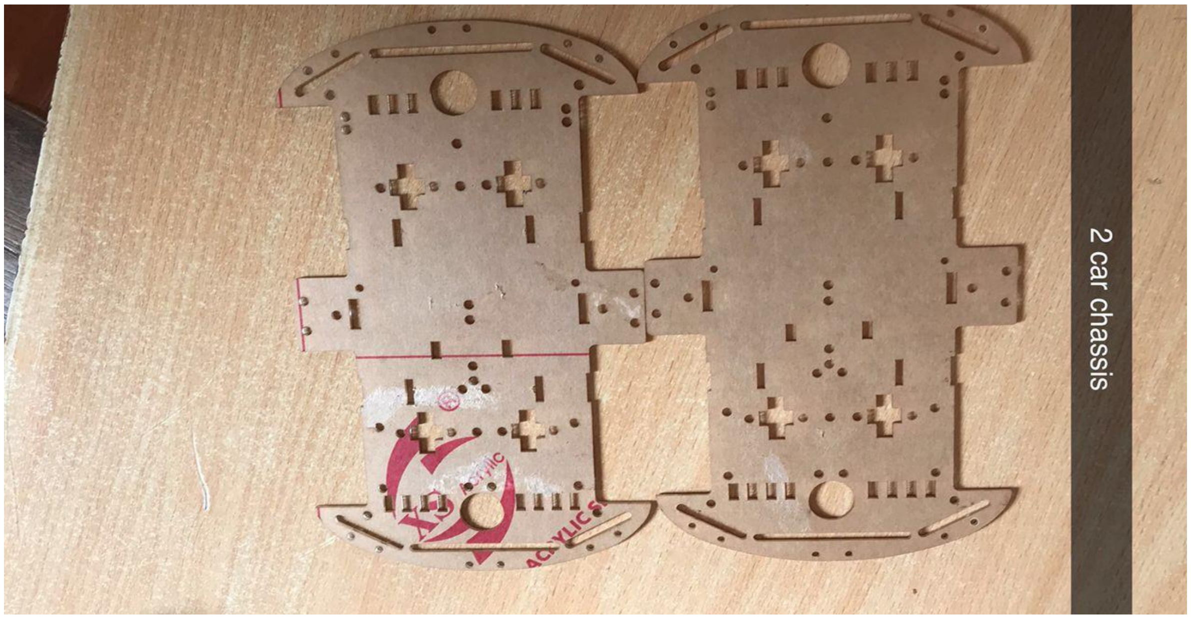
Objective:

To make the 4 wheel robotic car avoid obstacles (10 cm)

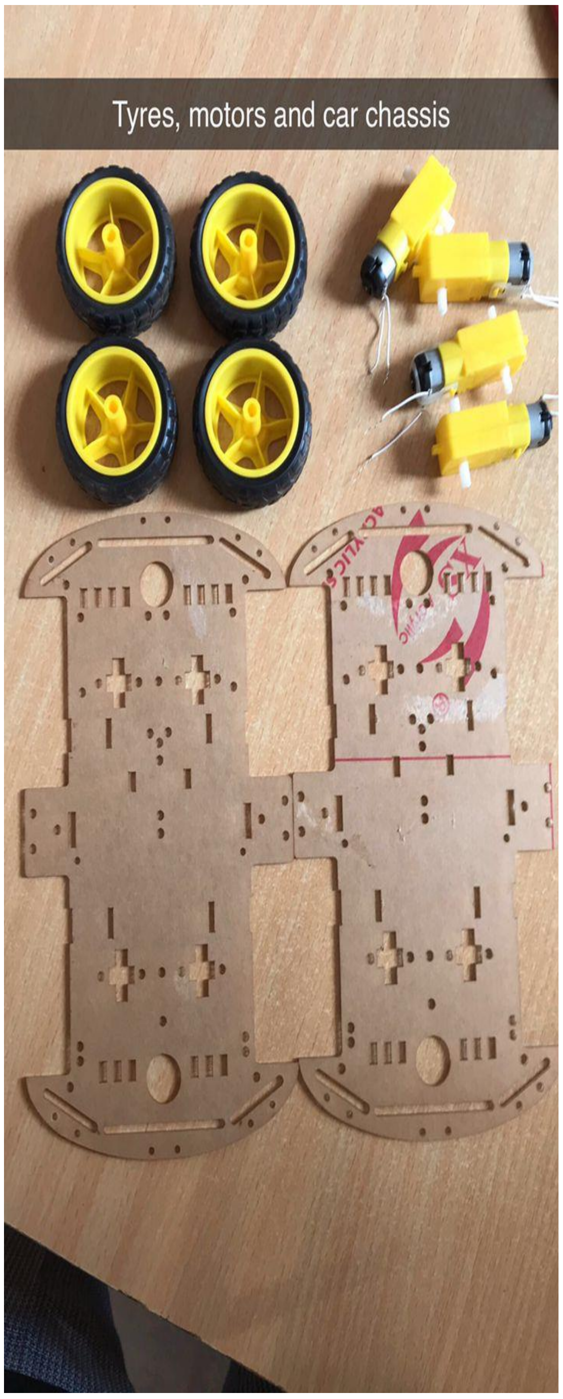
Tools:

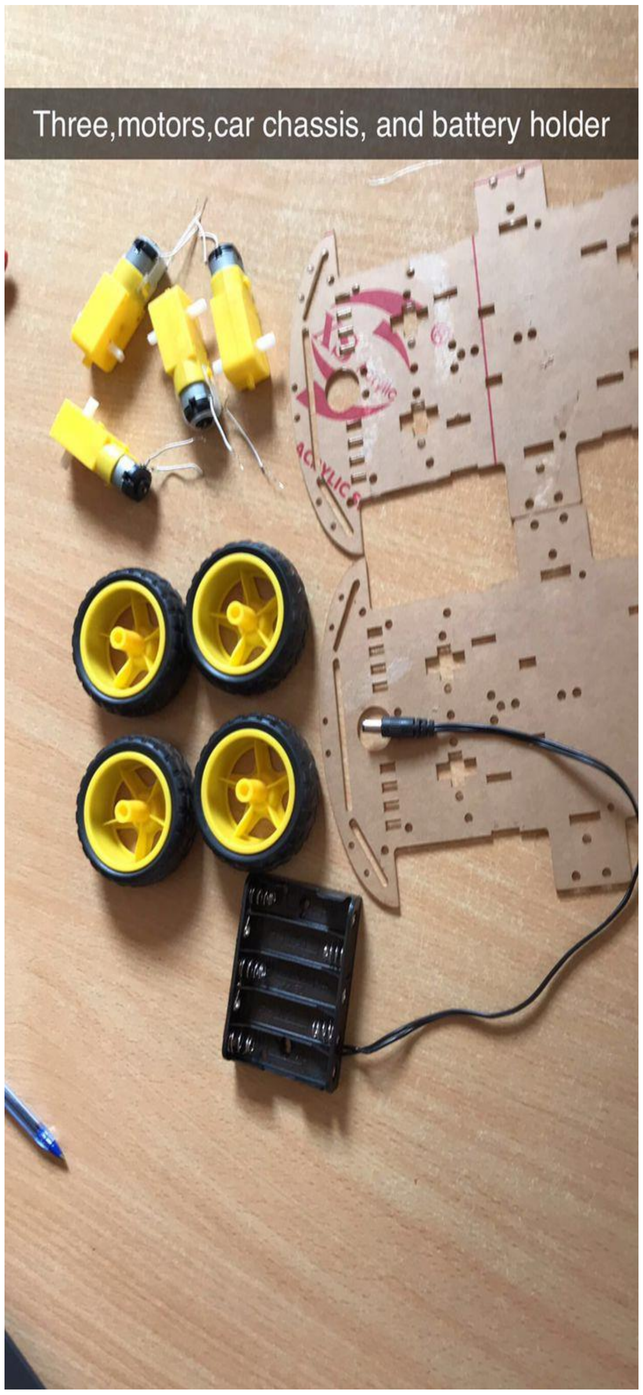
1 4 motors and 4 tyres
2 2 car chassis
3 Screws, Nuts and copper columns
4 Motor fixing brackets (8 pcs), speed measuring code disk 4 PCS
5 8 Long screws and Nuts
6 A 4-Way IR module and 4 IR sensor (obstacle avoidance sensor and channel tracking module.)
7 Ultrasonic sensor(distance sensor)
8 Motor driver
9 4 speed codes and 8 T-Brackets

FEATURES:



2 CAR CHASSIS



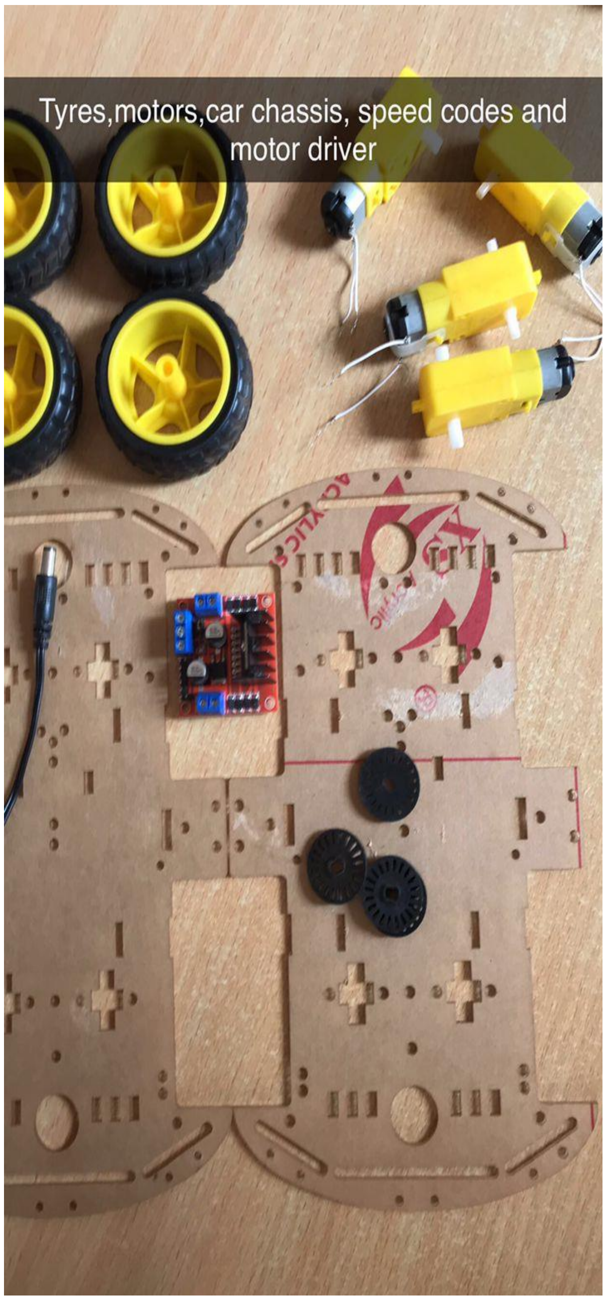


TYRES, MOTORS AND CAR CHASSIS

TYRES, CAR CHASSIS, MOTORS AND BATTERY HOLDER

FEMALE TO FEMALE AND MALE TO FEMALE WIRE JUMPERS

4 TYRES AND 2 CAR CHASSIS



TYRES, MOTORS,CAR CHASSIS,SPEED CODES AND MOTOR DRIVER

4 COPPER COLUMNS AND NUTS

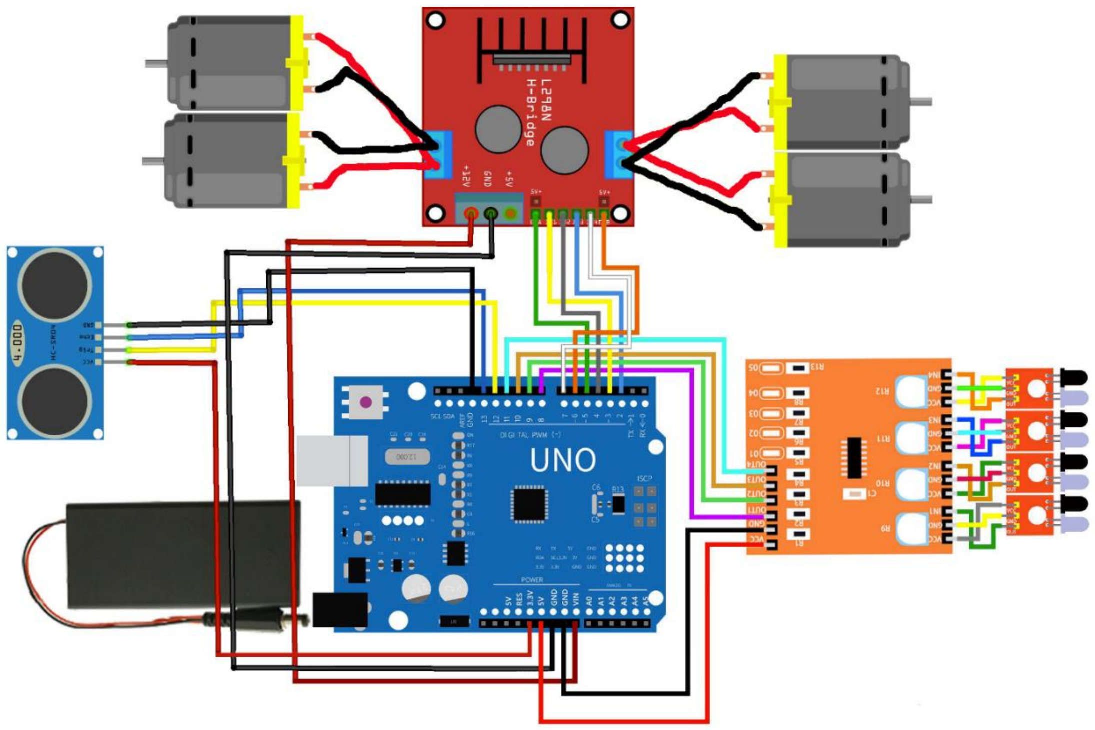
4 TYRES AND 4 MOTORS

8 LONG SCREWS AND NUTS

4 SPEED CODES

4 SPEED CODES AND 8 T-BRACKETS

CIRCUIT BOARD:



ACTIVITY:

LINE TRACKING MODE CODE

```
//define L298n module IO Pin
int ENA = 5;
int IN1 = 3;
int IN2 = 4;

int ENB = 6;
int IN3 = 2;
int IN4 = 7;

#define MotorASpeed 120
#define MotorBSpeed 120

int Sensor1 = 0;
int Sensor4 = 0;

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(Sensor1, INPUT);
  pinMode(Sensor4, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(ENA, HIGH);
  digitalWrite(ENB, HIGH);

  Sensor1 = digitalRead(8);
  Sensor4 = digitalRead(11);

  if (Sensor1 == HIGH && Sensor4 == HIGH) //IR is on black line
  {
    //Stop both Motors
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    analogWrite (ENA, 0);
    analogWrite (ENB, 0);
    Serial.println("Both IR is on black line - stop");
  }
  else if (Sensor1 == LOW && Sensor4 == LOW) //IR not on black line
  {
    //Move both the Motors
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite (ENA, MotorASpeed);
    analogWrite (ENB, MotorBSpeed);
    Serial.println("IR not on black line - go foward");
  }

  else if (Sensor1 == LOW && Sensor4 == HIGH)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite (ENA, 255);
    analogWrite (ENB, 255);
    Serial.println("Move left");
  }

  else if (Sensor1 == HIGH && Sensor4 == LOW)
  {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite (ENA, 255);
    analogWrite (ENB, 255);
    Serial.println("Move right");
  }

  else
  {
    //Stop both the motors
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    analogWrite (ENA, 0);
    analogWrite (ENB, 0);
    Serial.println("stop");
  }
}
```

CODE TO MAKE THE ROBOT AVOID OBSTACLES

```
#include <AFMotor.h>
#define Trig 12
#define Echo 13
#define ENA 5
#define ENB 6
#define IN1 3
#define IN2 4
#define IN3 2
#define IN4 7
float cm; //Distance variable
float temp; //

void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
}

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

  temp = float(pulseIn(Echo, HIGH));
  cm = (temp * 17 ) / 1000;
  if (cm < 30 && cm > 10)
  {
    back();
    delay(500);
    Left();
    delay(200);
  }
  if (cm >= 30)
  {
    forward();
    delay(100);
  }


  if (cm < 10)
  {
    STOP();
  }
  Serial.print("Echo =");
  Serial.print(temp);
  Serial.print(" | | Distance = ");
  Serial.print(cm);
  Serial.println("cm");
  delay(100);
}

void forward() {
  analogWrite(ENA, 220);
  analogWrite(ENB, 220);

  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Forward");
}

void back() {
  analogWrite(ENA, 220);
  analogWrite(ENB, 220);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("Back");
}


void Left() {
  analogWrite(ENA, 220);
  analogWrite(ENB, 220);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Left");
}

void Right() {
  analogWrite(ENA, 220);
  analogWrite(ENB, 220);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("Right");
}

void STOP() {
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  Serial.println("STOP");
}
```

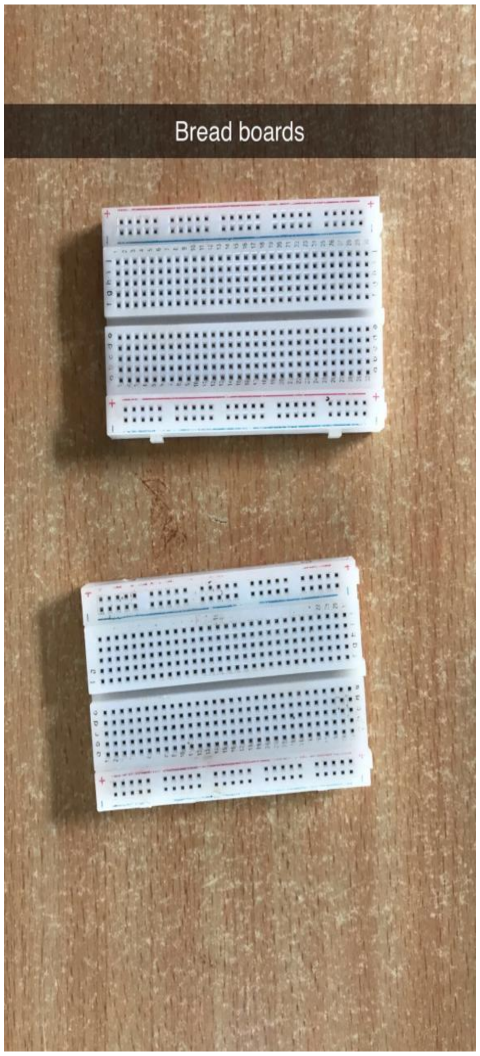
REPORT 6:

ACTIVITY 6:

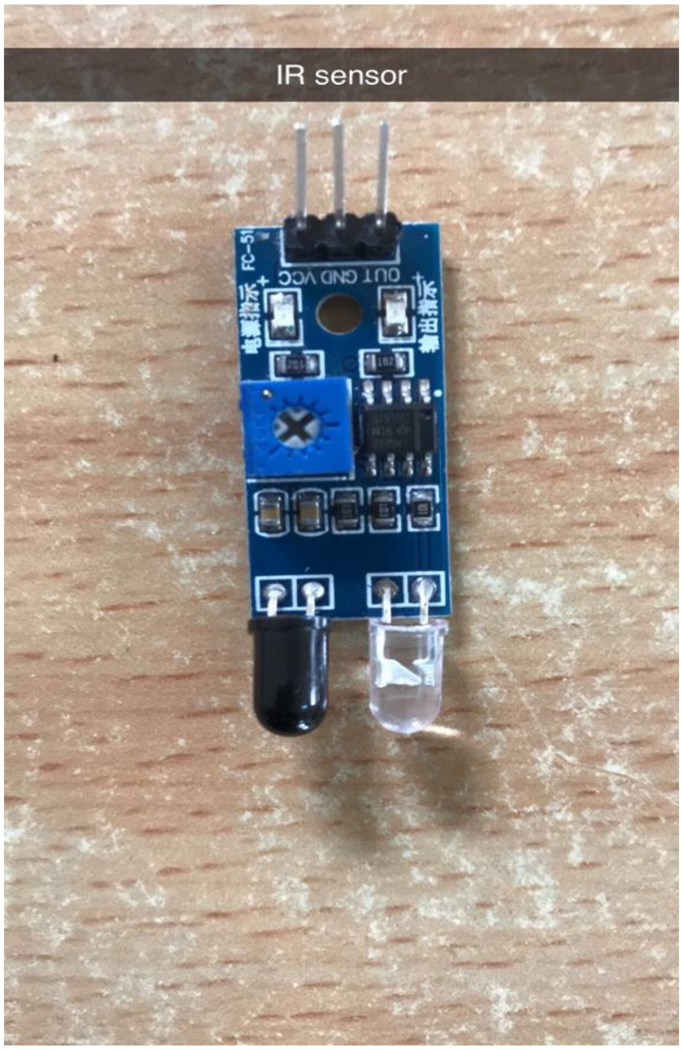
ROBOT MOVING ALONG THE EDGE OF A SURFACE

OBJECTIVE:

To move the 2 wheel robot along the edge of a surface using the IR sensor to detect edge of the surface
TOOLS USED: Sevor motors, wires(male to male and female to male), Arduino UNO R3, IR sensor, Breadboards, and a USB cable.

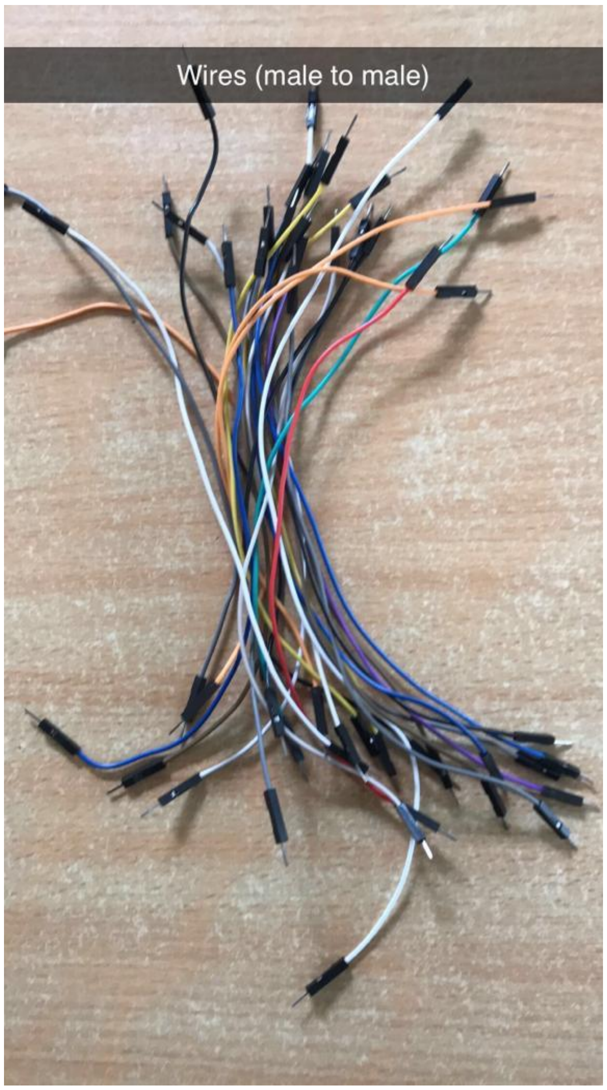


BREAD BOARDS

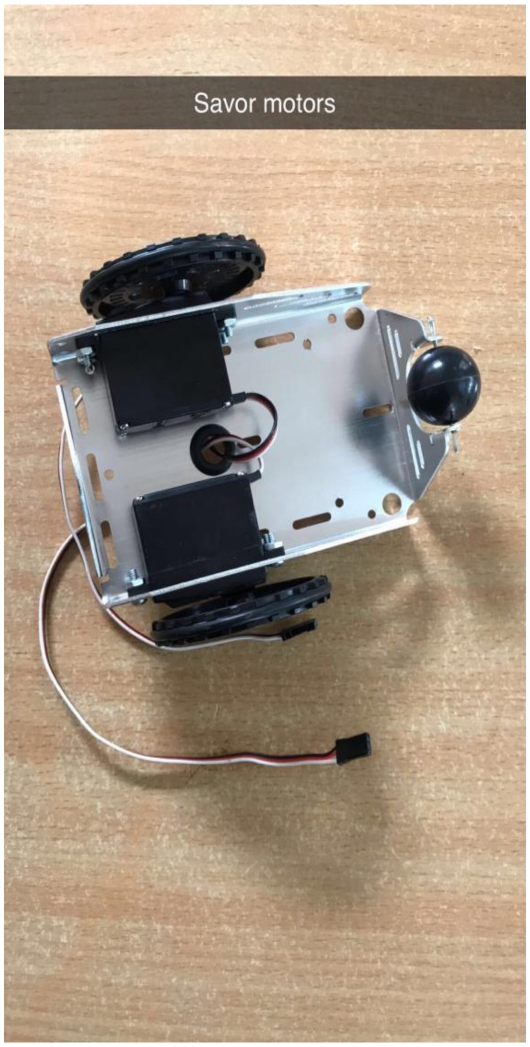


IR SENSOR

ARDUINO UNO R3



WIRES(MALE TO MALE)



SERVO MOTOR ARDUINO AND USB CABLE

CODE FOR MOVING ALONG THE EDGE ASSIGNMENT:

```
#include <Servo.h>
Servo myservo1,myservo2;
int sensor_pin=4;
int val;
void setup() {
  // put your setup code here, to run once:
 pinMode(sensor_pin,INPUT);


 myservo1.attach(9);
 myservo2.attach(10);
}

void loop() {
  // put your main code here, to run repeatedly:
 val=digitalRead(sensor_pin);
 if(val==0)
 {
  myservo1.write(0);
  myservo2.write(270);

}

 if(val==1)
 {
   myservo1.write(0);
   myservo2.write(0);
}
```

REPORT 7: REPORT ON ROBOT CAR MOVING ALONG THE WALK

Objective**: To make the robot move along the wall**

Tools used: **Servor motors, wires, Arduino UNO R3, breadboards, USB cable and Power bank.**

Code:

```
const int trigPin1 = A0;
const int echoPin1 = A1;
const int trigPin2 = A2;
const int echoPin2 = A3;
const int trigPin3 = A4;
const int echoPin3 = A5;

const int in1 = 7;
const int in2 = 6;
const int in3 = 5;
const int in4 = 4;
const int enA = 10;
const int enB = 9;

#define PWM 200
#define DIS 15

void setup()
{
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);

  pinMode (in1, OUTPUT);
  pinMode (in2, OUTPUT);
  pinMode (in3, OUTPUT);
  pinMode (in4, OUTPUT);
  pinMode (enA, OUTPUT);
  pinMode (enB, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Sensor 1=");
  Serial.println(FrontSensor());
  Serial.println("Sensor 2=");
  Serial.println(RightSensor());
  Serial.println("Sensor 3=");
  Serial.println(LeftSensor());
void forward ()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enA, PWM);
  analogWrite(enB, PWM);
}

void turn_left ()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enA, PWM);
  analogWrite(enB, PWM);
}

void turn_right ()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enA, PWM);
  analogWrite(enB, PWM);
}

void reverse ()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enA, PWM);
  analogWrite(enB, PWM);
}

void stop()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enA, LOW);
  analogWrite(enB, LOW);
}
```