University of Northampton

الجـامعـة الخـليجـية
GULF UNIVERSITY

College of Engineering

Department: **Electrical and Electronic Engineering**

Course Title: 24/25 - Microprocessor Systems

Course Code: CSY2015-PGU-P1

Student ID: 23855994

Student Name: Sayed Meshal Alawi

Instructor Name: Dr. Osama Al-Rawi

University of Northampton

الجـامعـة الخـليجـية
GULF UNIVERSITY

Part one- radar system

YouTube link

https://youtube.com/shorts/VAezIqb8nu8?feature=share

Introduction:

I have created a system that utilizes radar for the purpose of tracking objects. The radar system employs ultrasonic or electromagnetic waves, and it can sense the presence of an object as well as the distance from it hence this gives it potential in many areas like robotics, security systems and traffic control. For example, perimeter intrusion detection and surveillance can employ radar technology, using radar systems deployed for Security Systems. Motion detecting and warning users on possible threats are some of its functions.

Advantages:

Radar systems are advantageous for many applications particularly because they can remotely sense or detect an object. Here are some out of many benefits of the radar systems Uses radio waves to determine the range, angle of objects and used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles and terrain.

Disadvantages:

The disadvantages of this technology are Reduced detection range and does not recognize the color of the targets.

Future development:

The targets color can be identified, and the radar's detection range can be enhanced.

Components list:

- Arduino UNO
- Ultrasonic Sensor
- Servo Motor sg90
- I2C Lcd
- Jumper wire
- Breadboard

Objective:

I chose the radar system because it has so much variety and it can be improved more.

Conclusion:

The radar functions which consist of sound waves, or electromagnetic waves for that matter, can be used for various purpose including range measurement and detection of the existence of any object.

Flow chart for part one:

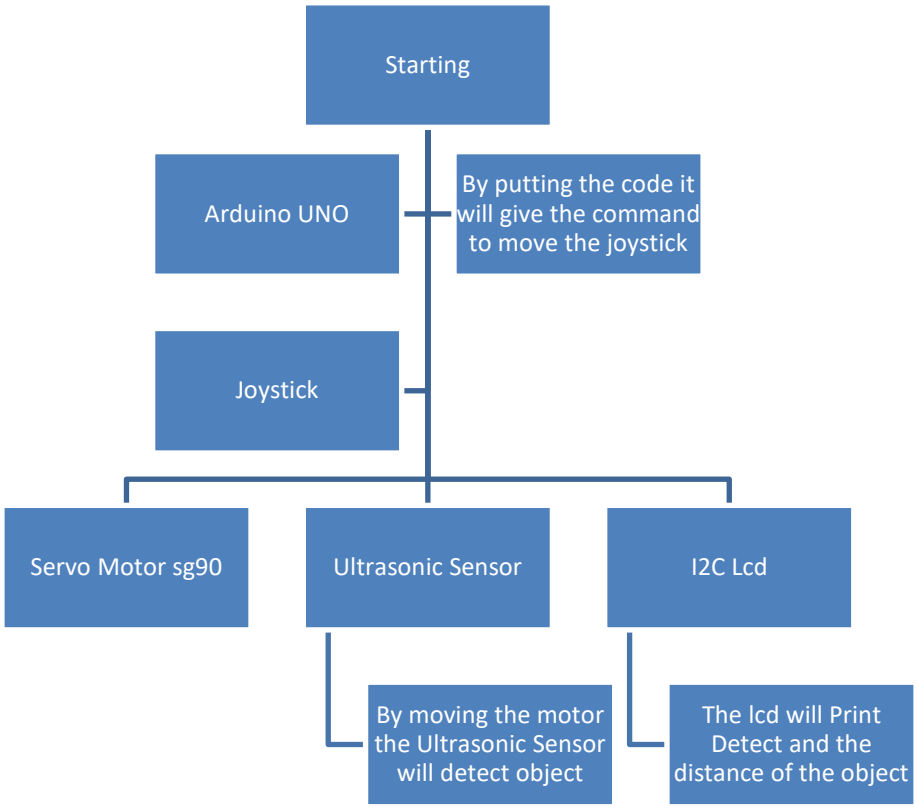
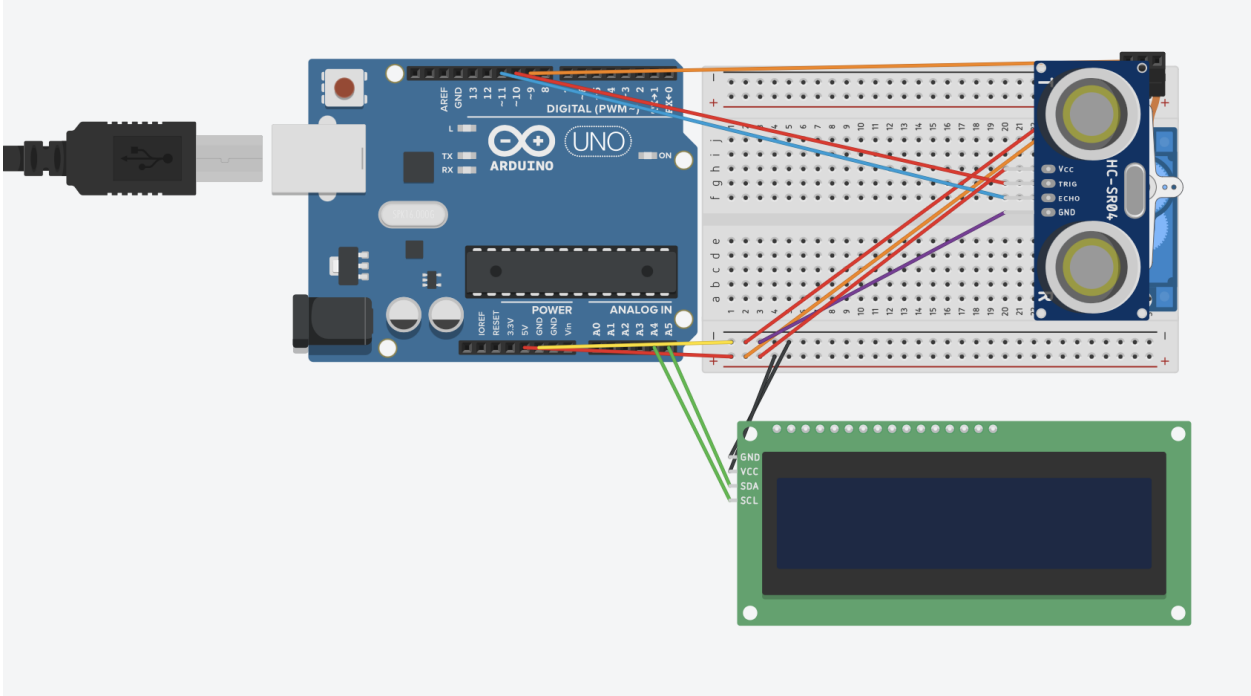


Figure 1.



Unfortunately, I didn't find the joystick in the tinker cad that's why I didn't put one

Code:

```
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal_I2C library

#include <Servo.h>


const int trigPin = 10;

const int echoPin = 11;

long duration;

int distance;


Servo myServo;

const int joyStickPin = A0; // Analog pin connected to joystick output

int val;
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // Initialize the LCD with the I2C address and dimensions

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output for the ultrasonic sensor
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input for the ultrasonic sensor
  Serial.begin(9600); // Initializes the serial communication

  myServo.attach(9); // Attach the servo motor

  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  lcd.setCursor(0, 0); // Set the cursor to the first row
  lcd.print("Distance: "); // Initial LCD message
}

void loop() {
  // Ultrasonic sensor part
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
```

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

  // Print distance on LCD
  lcd.setCursor(9, 0); // Set the cursor to the distance value position
  lcd.print(distance); // Print the distance on the LCD

  // Joystick control for the servo motor
  val = analogRead(joyStickPin);
  val = map(val, 0, 1023, 15, 165); // Map joystick values to servo angles
  myServo.write(val);

  // Check if distance is within a certain range to trigger the "detect" message
  if (distance < 50) {
    lcd.setCursor(0, 1); // Set the cursor to the second row
    lcd.print("Detect"); // Print "Detect" on the LCD
  } else {
    lcd.setCursor(0, 1); // Set the cursor to the second row
    lcd.print("          "); // Clear the second row if no object detected
  }

  delay(100);
}
```

University of Northampton

الجـامعـة الخـليجـية
GULF UNIVERSITY

Part two- RGB Floodlight

YouTube link

https://youtube.com/shorts/rwBGs7HueJI?feature=share

Introduction:

The second stage of the project required us to Create a prototype for a home security floodlight. This project makes use of an RGB LED floodlight which can be controlled with the help of Arduino board and an IR remote. A remote control allows you to manage the RGB Floodlight functions. RGB Floodlight displays different colors when commanded with specific IR commands. The Arduino is programmed to receive commands such as red, green, blue and white from the IR remote which it uses to control the corresponding colors on the LED. All that is required is to send a few signal commands through the IR remote to adjust the effects of the RGB floodlight.

Disadvantages:

The project is very complex

Conclusion:

In summary, this project demonstrates the possibility of integrating devices such as Arduino, IR remote controls and RGB floodlights together to build interactive lighting systems with possible use in areas of home security where the effect of lighting is both useful to the user.

Components list:

- Arduino uno
- RGB led light
- IR remote
- IR receiver
- LDR
- Ultrasonic sensor
- Jumper wire
- Breadboard

Flow chart for part two:

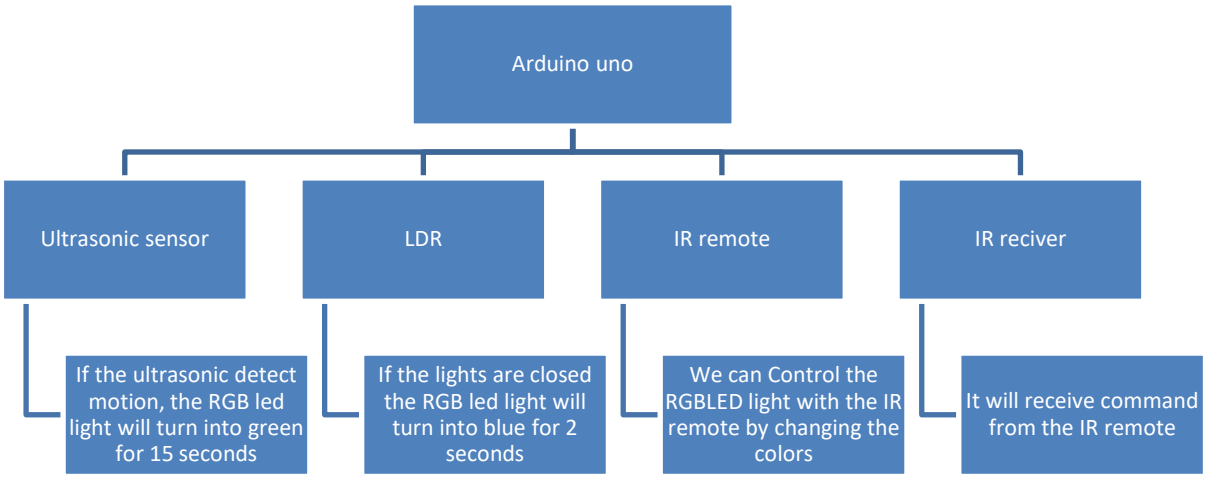
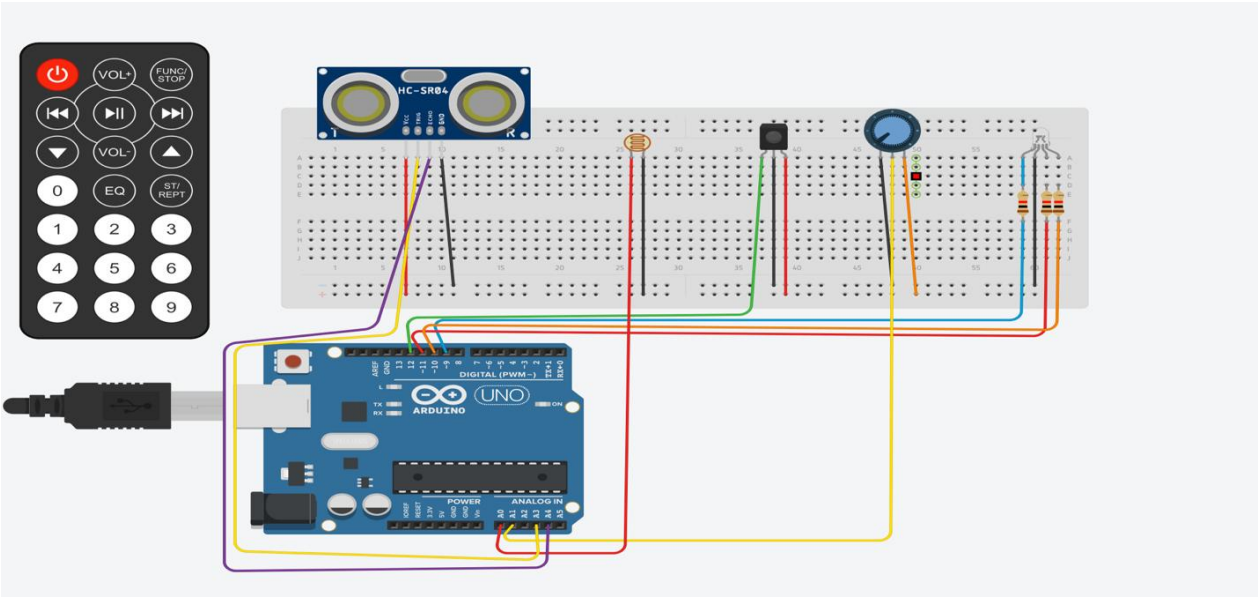


Figure 2.



Code:

```
#include <IRremote.hpp>

// Pin Setup
const int redLED = 8;
const int greenLED = 9;
const int blueLED = 10;
const int ultrasonicTrigger = 3;
const int ultrasonicEcho = 4;
const int lightSensor = A0;
const int potentiometer = A1;
const int irSensor = 12;

// Constants
const int lightThreshold = 9;
const int proximityLimit = 50;
const unsigned long rgbTimeout = 15000;
const unsigned long minLightActive = 2000;
const unsigned long maxLightActive = 4000;

// State Variables
bool rgbState = false;
unsigned long rgbTimer = 0;
int ledIntensity = 255;

// IR Commands
const uint32_t CMD_RED = 10;
```

```
const uint32_t CMD_GREEN = 22;
const uint32_t CMD_BLUE = 92;
const uint32_t CMD_WHITE = 6;
const uint32_t CMD_SLOW_CYCLE = 67;
const uint32_t CMD_FAST_CYCLE = 74;

// Color Table
int rgbValues[][3] = {
  {255, 0, 0},    // Red
  {0, 255, 0},    // Green
  {0, 0, 255},    // Blue
  {255, 255, 255} // White
};
int currentColorIndex = 0;

IRrecv irReceiver(irSensor);
decode_results irData;

void setup() {
  Serial.begin(9600);
  IrReceiver.begin(irSensor, ENABLE_LED_FEEDBACK);

  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(ultrasonicTrigger, OUTPUT);
  pinMode(ultrasonicEcho, INPUT);
```

```
  pinMode(lightSensor, INPUT);
  pinMode(potentiometer, INPUT);



  setLEDColor(0, 0, 0);
  Serial.println("System Initialized");
}

void loop() {
  adjustIntensity();

  if (IrReceiver.decode()) {
    handleIRInput(IrReceiver.decodedIRData.command);
    IrReceiver.resume();
  }

  checkUltrasonic();
  checkLDR();

  if (rgbState && millis() > rgbTimer) {
    rgbState = false;
    setLEDColor(0, 0, 0);
  }

  delay(100);
}
```

```
void adjustIntensity() {
  int potValue = analogRead(potentiometer);
  ledIntensity = map(potValue, 0, 1023, 0, 255);
}

void checkUltrasonic() {
  float distance = getProximity();
  if (distance > 0 && distance <= proximityLimit && !rgbState) {
    rgbState = true;
    rgbTimer = millis() + rgbTimeout;
    setLEDColor(ledIntensity, 0, 0); // Red
    Serial.println("Motion Detected. RGB Activated.");
  }
}

void checkLDR() {
  int lightReading = analogRead(lightSensor);
  if (lightReading < lightThreshold && !rgbState) {
    rgbState = true;
    rgbTimer = millis() + random(minLightActive, maxLightActive);
    setLEDColor(0, ledIntensity, ledIntensity); // Cyan
    Serial.println("Dark Environment Detected.");
  }
}

float getProximity() {
  digitalWrite(ultrasonicTrigger, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(ultrasonicTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(ultrasonicTrigger, LOW);

  long duration = pulseIn(ultrasonicEcho, HIGH);
  return (duration * 0.034) / 2;
}

void handleIRInput(uint32_t command) {
  switch (command) {
    case CMD_RED:
      setLEDColor(ledIntensity, 0, 0);
      break;
    case CMD_GREEN:
      setLEDColor(0, ledIntensity, 0);
      break;
    case CMD_BLUE:
      setLEDColor(0, 0, ledIntensity);
      break;
    case CMD_WHITE:
      setLEDColor(ledIntensity, ledIntensity, ledIntensity);
      break;
    case CMD_SLOW_CYCLE:
      cycleLEDColors(3000);
      break;
    case CMD_FAST_CYCLE:
```

```
      cycleLEDColors(1000);
      break;
    default:
      Serial.println("Unrecognized Command");
      break;
  }
}

void setLEDColor(int red, int green, int blue) {
  analogWrite(redLED, red);
  analogWrite(greenLED, green);
  analogWrite(blueLED, blue);
}

void cycleLEDColors(unsigned long delayTime) {
  for (int i = 0; i < 4; i++) {
    setLEDColor(rgbValues[i][0], rgbValues[i][1], rgbValues[i][2]);
    delay(delayTime);
  }
}
```