MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

Arudino and Raspberry Pi

Assignment 1

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To learn about Arduino UNO, and Raspberry Pi Model 3B in detail and thier applications in IoT.

2 Objectives

- To learn about Arduino UNO and Raspberry Pi Model 3B.
- To learn about the applications of Arduino UNO and Raspberry Pi Model 3B in IoT.

3 Equipments Required

1. Raspberry Pi 3 Model
2. Arduino Uno
3. Breadboard
4. Jumper wires
5. LED
6. Resistor

4 Theory

4.1 Raspberry Pi Model 3B

Raspberry Pi Model 3B is a popular single-board computer developed by the Raspberry Pi Foundation. It is a successor to the Raspberry Pi 2 Model B and features a quad-core ARM Cortex-A53 CPU with 1GB of RAM. The board has a variety of input/output options, including HDMI, USB, Ethernet, and a 40-pin GPIO (General Purpose Input/Output) header.

It can run various operating systems, including Raspbian, Ubuntu, and Windows 10 IoT Core. Raspberry Pi 3B is used in a variety of projects, including home automation, robotics, and media centers.

4.2 Arduino Uno

On the other hand, Arduino Uno is a microcontroller board based on the ATmega328P microcontroller. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, and an ICSP header.

It can be powered using a USB cable or an external power source. Arduino Uno can be programmed using the Arduino IDE, which supports a variety of programming languages, including C and C++.

It is commonly used in various projects, including home automation, robotics, and IoT applications.

4.3 Difference between Raspberry Pi 3B and Arduino Uno

One of the main differences between Raspberry Pi 3B and Arduino Uno is their architecture. Raspberry Pi is a full-fledged computer that can run an operating system, while Arduino Uno is a microcontroller board that runs a single program.

This makes Raspberry Pi more suitable for projects that require more processing power, such as media centers or web servers. Arduino Uno, on the other hand, is more suitable for projects that require real-time control, such as robotics or home automation.

Another difference between the two is their input/output capabilities. Raspberry Pi 3B has a variety of input/output options, including HDMI, USB, Ethernet, and a 40-pin GPIO header, making it suitable for a wide range of projects. Arduino Uno, on the other hand, has a smaller number of input/output pins, making it suitable for simpler projects that require fewer inputs/outputs.

In terms of the Internet of Things (IoT), both Raspberry Pi 3B and Arduino Uno can be used to create IoT applications. Raspberry Pi 3B can be used as a web server, which can be accessed remotely over the Internet.

It can also be used to interface with various sensors and actuators, making it suitable for IoT applications. Arduino Uno, on the other hand, can be used to create standalone IoT devices that can communicate with other devices over the Internet.

It can also be used to interface with various sensors and actuators, making it suitable for a wide range of IoT applications.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Arduino IDE, and Thonny on Pi
Compilers : g++ and gcc on linux for C++, Python 3.10 on Pi

6 Arduino Uno

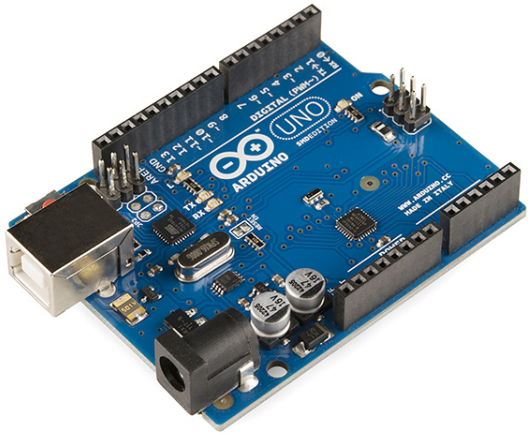

Figure 1: Tinkercad Circuit

7 Arduino Uno Pin Diagram

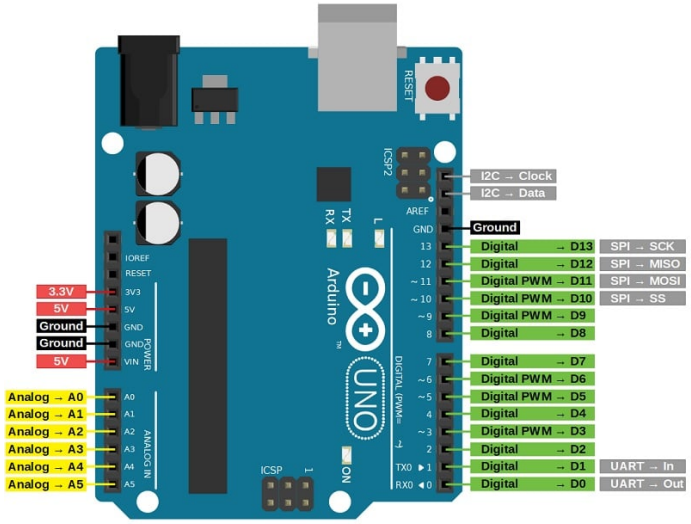

Figure 2: Arduino Uno Pin Diagram

8 Raspberry Pi 3B

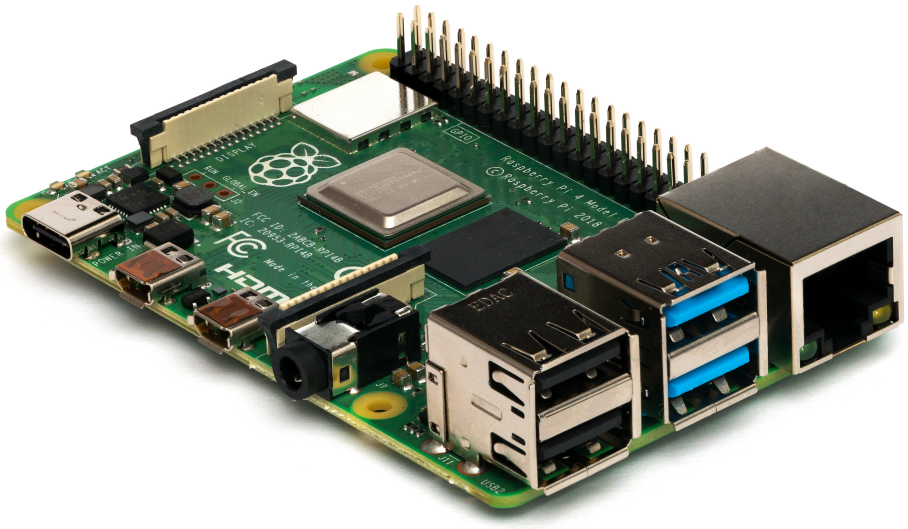

Figure 3: Circuit Diagram

9 Raspberry Pi 3B Pin Diagram

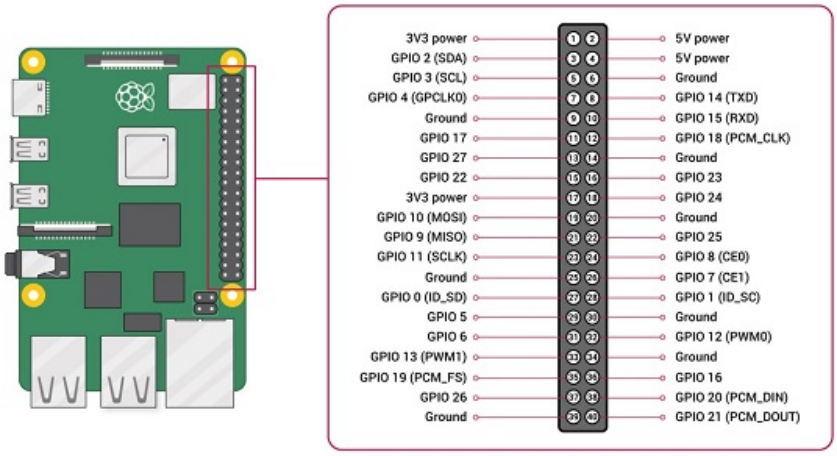

Figure 4: Circuit Diagram

10 Conclusion

Thus, we have successfully interfaced Temperature Sensor with Arduino Uno and displayed the output on the Serial Monitor.

11 FAQ

1. Arduino Uno R3 (Code: U1)

    - Type: Microcontroller board
    - Digital pins: 14
    - Analog input pins: 6
    - Operating voltage: 5V
    - Input voltage (recommended): 7-12V
    - Input voltage (limits): 6-20V
    - Flash memory: 32 KB (ATmega328P microcontroller)
    - SRAM: 2 KB (ATmega328P microcontroller)
    - EEPROM: 1 KB (ATmega328P microcontroller)
    - Clock speed: 16 MHz (ATmega328P microcontroller)

2. Raspberry Pi Model 3 B:
   The Raspberry Pi has a total of 40 pins, including 26 GPIO (General Purpose Input/Output) pins, 3.3V and 5V power pins, and ground pins. The pinout diagram for the Raspberry Pi 3 B can be found on the official Raspberry Pi website.

    - Pin 1: 3.3V
    - Pin 2: 5V
    - Pin 3: GPIO 2
    - Pin 4: 5V
    - Pin 5: GPIO 3
    - Pin 6: Ground
    - Pin 7: GPIO 4
    - Pin 8: GPIO 14
    - Pin 9: Ground
    - Pin 10: GPIO 15
    - Pin 11: GPIO 17
    - Pin 12: GPIO 18
    - Pin 13: GPIO 27
    - Pin 14: Ground
    - Pin 15: GPIO 22
    - Pin 16: GPIO 23
    - Pin 17: 3.3V
    - Pin 18: GPIO 24
    - Pin 19: GPIO 10
    - Pin 20: Ground
    - Pin 21: GPIO 9

- Pin 22: GPIO 25
- Pin 23: GPIO 11
- Pin 24: GPIO 8
- Pin 25: Ground
- Pin 26: GPIO 7
- Pin 27: ID SD
- Pin 28: ID SC
- Pin 29: GPIO 5
- Pin 30: Ground
- Pin 31: GPIO 6
- Pin 32: GPIO 12
- Pin 33: GPIO 13
- Pin 34: Ground
- Pin 35: GPIO 19
- Pin 36: GPIO 16
- Pin 37: GPIO 26
- Pin 38: GPIO 20
- Pin 39: Ground
- Pin 40: GPIO 21

MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

Arudino with Sensors and Actuators

Assignment 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To interface following Sensors such as Temperature or Ultrasonic or IR or any other sensor with Arduino Uno and display the output on the Serial Monitor.

2 Objectives

- To interface Temperature Sensor with Arduino Uno and display the output on the Serial Monitor.
- To learn how to use Arduino Uno.
- To learn about Actuators and Sensors.

3 Equipments Required

Name	Quantity	Component
U1	1	Arduino Uno R3
U5	1	Temperature Sensor [TMP36]
PIEZO1	1	Piezo
D1	1	Red LED
MFAN	1	DC Motor
R1	1	1 kΩ Resistor

4 Theory

In this assignment, we will be interfacing a temperature sensor with Arduino Uno and displaying the output on the serial monitor. The temperature sensor used in this project is an analog sensor, which measures temperature by outputting a voltage proportional to the temperature. The TMP36 temperature sensor is a popular choice due to its accuracy and low cost.

The Arduino Uno is a microcontroller board that is commonly used for prototyping and educational purposes. It has 14 digital input/output pins and 6 analog input pins, and is powered by a 5V supply. The board is programmed using the Arduino programming language, which is a simplified version of C++.

To interface the temperature sensor with the Arduino, we connect the output pin of the sensor to one of the analog input pins on the Arduino, and connect the ground and power pins to the appropriate pins on the Arduino. We then read the analog voltage from the sensor using the analogRead() function in the Arduino code.

The output of the temperature sensor is displayed on the serial monitor, which is a useful tool for debugging and monitoring the output of the Arduino. The serial monitor allows us to view the output in real-time and make any necessary adjustments to the code or hardware.

In the context of the Internet of Things (IoT), this project can be extended to include wireless communication, allowing the temperature readings to be monitored and analyzed remotely. For

example, the Arduino could be connected to a Wi-Fi module or a cellular module, and the temperature readings could be sent to a cloud-based platform for analysis and visualization. This could be useful in applications such as environmental monitoring or industrial automation.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Arduino IDE
Compilers : g++ and gcc on linux for C++

6 Circuit

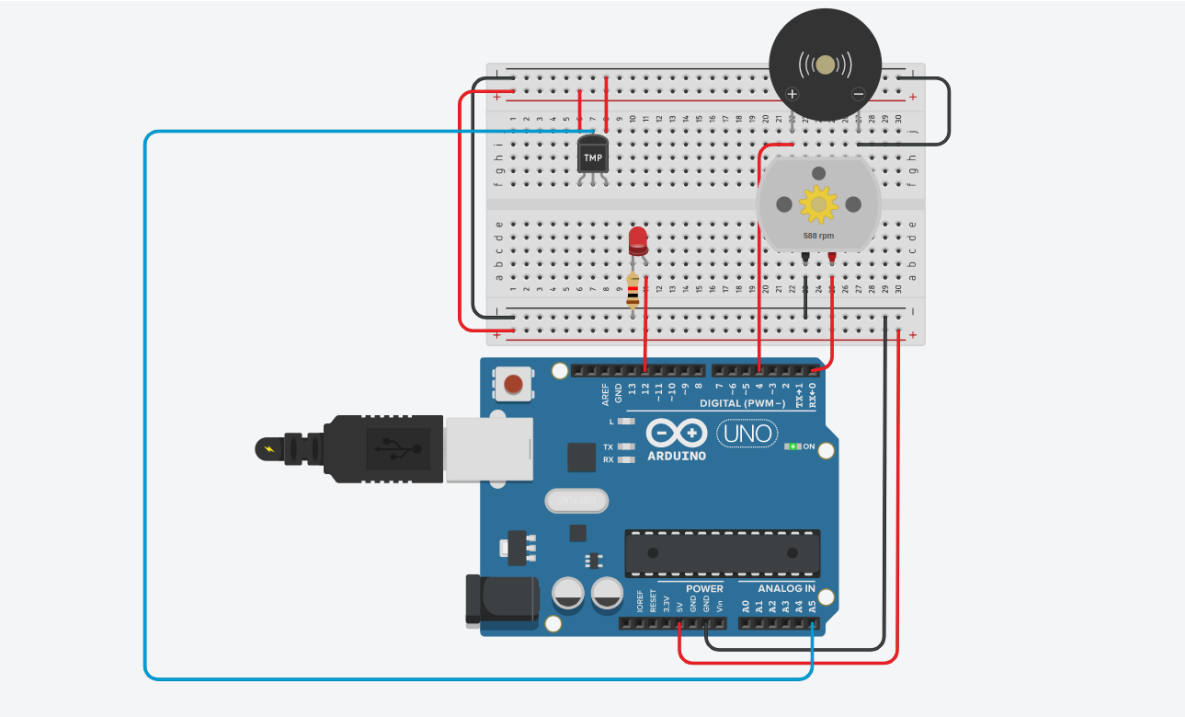

Figure 1: Tinkercad Circuit

7 Circuit Diagram

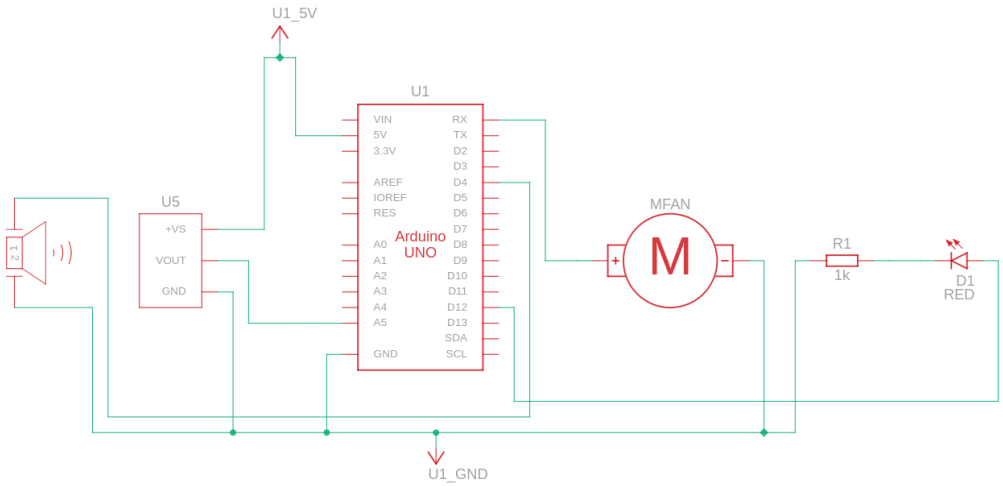

Figure 2: Circuit Diagram

8 Input

Input from the Temperature Sensor TMP36

9 Output

```
Temperature is too much
153
Temperature is too much
153
27
27
20
20
```

10 Code

```
1
2  #define LED_PIN 12
3  #define MOTOR_PIN 0
4  #define BUZZER_PIN 4
5  int analogPin = A5;
6  int val = 0;
7
8
9  void setup() {
10   Serial.begin(9600);  //  setup serial
```

```
  pinMode(LED_PIN, OUTPUT);
  pinMode(MOTOR_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val); // debug value

  if(val > 100){
    Serial.println("Temperature is too much");
  digitalWrite(0, HIGH);
    digitalWrite(LED_PIN, HIGH);
  tone(BUZZER_PIN, val, 1000);
  }
}
```

11 Conclusion

Thus, we have successfully interfaced Temperature Sensor with Arduino Uno and displayed the output on the Serial Monitor.

12 FAQ

1. Arduino Uno R3 (Code: U1)
    - Type: Microcontroller board
    - Digital pins: 14
    - Analog input pins: 6
    - Operating voltage: 5V
    - Input voltage (recommended): 7-12V
    - Input voltage (limits): 6-20V
    - Flash memory: 32 KB (ATmega328P microcontroller)
    - SRAM: 2 KB (ATmega328P microcontroller)
    - EEPROM: 1 KB (ATmega328P microcontroller)
    - Clock speed: 16 MHz (ATmega328P microcontroller)
2. Temperature Sensor [TMP36] (Code: U2)
    - Type: Analog sensor
    - Pin 1: Output voltage (analog)
    - Pin 2: Ground
    - Pin 3: Input voltage (3.3V - 5V)
    - Temperature range: -40°C to +125°C
    - Output voltage range: 0V to 1.75V (at 25°C)
3. Buzzer (Code: U4)
    - Type: Electromechanical transducer
    - Operating voltage: 5V to 12V
    - Sound frequency: 2kHz to 4kHz
    - Sound pressure level: 80dB to 110dB
    - Pin 1: Positive terminal (+)
    - Pin 2: Negative terminal (-)
4. DC Motor:
    - Type: Electromechanical motor
    - Operating voltage: Varies depending on the motor specifications, typically 3-12V
    - Maximum voltage: Varies depending on the motor specifications, typically 6-24V
    - Maximum current: Varies depending on the motor specifications, typically 100mA - 1A
    - Control type: Typically controlled using pulse width modulation (PWM)
    - Speed range: Varies depending on the motor specifications, typically 1000-5000 RPM
    - Direction of rotation: Can be reversed by reversing the polarity of the power supply
    - Shaft diameter: Varies depending on the motor specifications, typically 2-5mm
    - Stall torque: Varies depending on the motor specifications, typically 0.1-1 Nm
    - Gearbox: Some DC motors come with a gearbox for increased torque or reduced spee

MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

Interface of Servo Motor like Actuators with Development Boards

---

Assignment 3

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To interface simple actuators such as DC Motor with Raspberry Pi/ ESP8266 boards / Beagle bone board/ Tinker CAD Arduino Uno.

2 Objectives

- To understand actuators interfacing with development boards
- Servo Motor control using LN298 motor driver and Arduino Uno

3 Component List

Name	Quantity	Component
U1	1	Arduino Uno R3
DIST1	1	Ultrasonic Distance Sensor
SERVO1	1	Positional Micro Servo

4 Theory

4.1 Stepper Motor

A stepper motor is a type of electric motor that rotates in small, precise steps. Stepper motors are commonly used in applications that require precise positioning, such as robotics, CNC machines, and 3D printers. Stepper motors have multiple coils that are energized in a specific sequence to produce precise steps of rotation.

The rotation of the stepper motor is controlled by a series of pulses sent to the motor driver, which energizes the coils in the correct sequence.

4.2 DC Motor

A DC (direct current) motor is a type of electric motor that uses a magnetic field to produce rotational motion. DC motors typically have two terminals that are connected to a DC power source.

When a current is applied to the motor, it generates a magnetic field that interacts with the magnetic field of the stator, causing the rotor to rotate. DC motors are commonly used in applications that require high torque and low speed, such as robotics, conveyor systems, and electric vehicles.

4.3 Servo Motor

A servo motor is a type of electric motor that uses feedback to control the position of the output shaft. Servo motors are commonly used in applications that require precise control of position and speed, such as robotics, RC cars, and model airplanes.

Servo motors have a built-in feedback mechanism that detects the position of the output shaft and adjusts the motor's operation accordingly. Servo motors typically have a limited range of motion (usually less than 180 degrees) and are used for applications that require precise control of motion.

4.4 LN298 Motor Driver

1. Dual H-bridge configuration: The L298 consists of two H-bridge circuits, which allows it to drive two DC motors or one bipolar stepper motor.
2. High current capacity: The L298 can handle a maximum current of up to 2 amps per channel, which makes it suitable for driving high-power motors.
3. Low voltage drop: The L298 has a low voltage drop, which means that it can operate with low voltage power supplies.
4. Built-in protection features: The L298 includes built-in protection features such as thermal shutdown, overvoltage protection, and under-voltage lockout, which help to protect the device from damage.
5. TTL and CMOS compatible inputs: The L298 has TTL and CMOS compatible inputs, which allows it to be easily interfaced with microcontrollers and other digital circuits.
6. Wide operating voltage range: The L298 can operate over a wide voltage range, typically from 5 volts to 46 volts, which makes it suitable for a variety of applications.
7. Compact package: The L298 is available in a compact 15-pin package, which makes it easy to integrate into small electronic devices.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Arduino UNO
Compilers : C++ GNU Compiler

6 Circuit

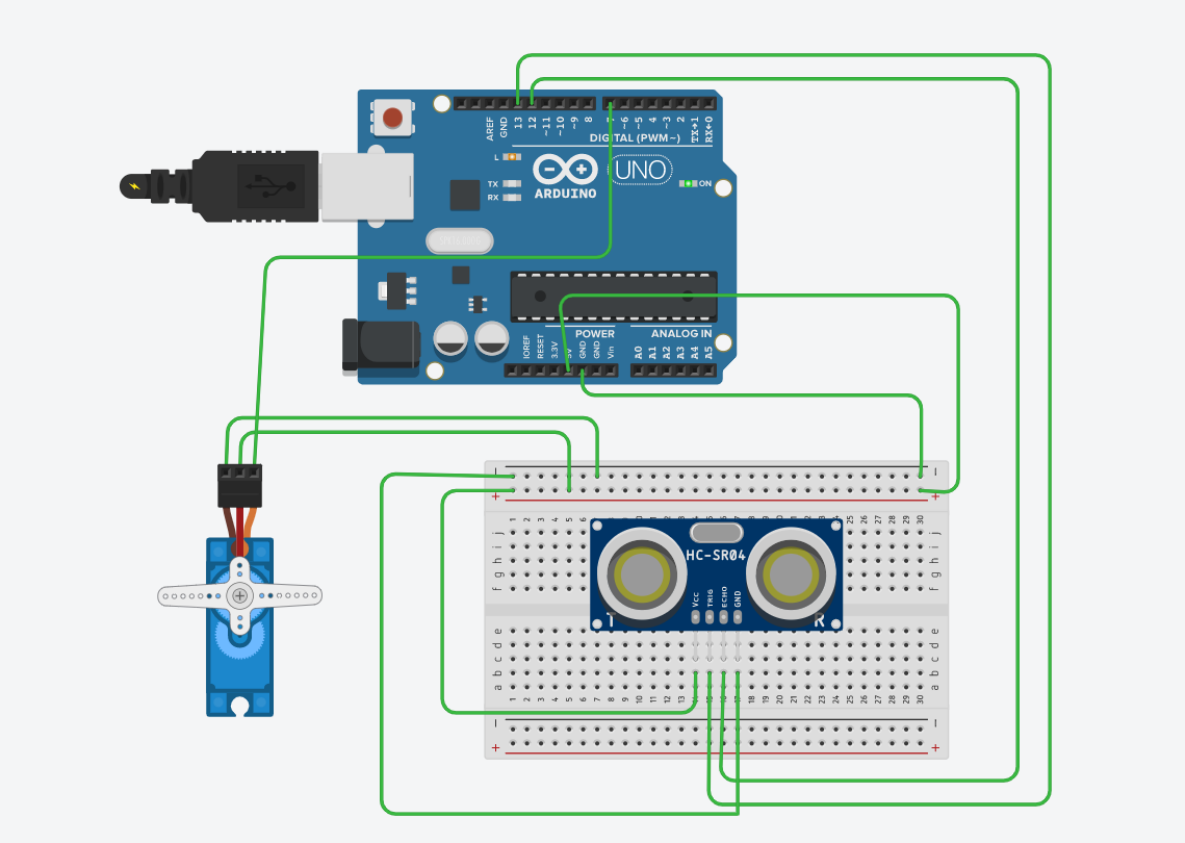

Figure 1: Circuit Diagram

7 Circuit Diagram

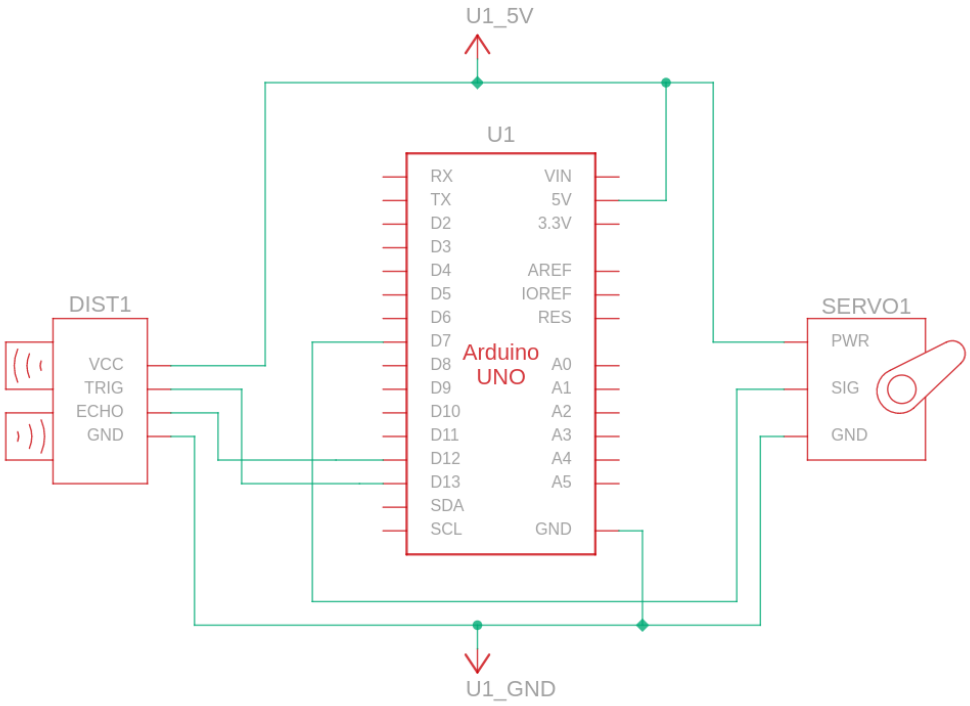

Figure 2: Circuit Diagram

8 Code

```
#include<Servo.h>
Servo srv;
#define maxdistance 100
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  pinMode(12, INPUT);
  srv.attach(7);

}

void loop()
{

  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  int d=pulseIn(12,HIGH);
  d=d/29/2;
  Serial.println(d);
```

```
24      if(d<=maxdistance)
25      {
26      srv.write(90);
27      delay(1000);
28      }
29      else
30      {
31      delay(1000);
32      srv.write(0);
33      }
34
35    }
```

9 Conclusion

Thus, we have successfully controlled the servo motor using the ultrasonic sensor and Arduino Uno.

10 FAQ

1. Arduino Uno R3 (Code: U1)
    - Type: Microcontroller board
    - Digital pins: 14
    - Analog input pins: 6
    - Operating voltage: 5V
    - Input voltage (recommended): 7-12V
    - Input voltage (limits): 6-20V
    - Flash memory: 32 KB (ATmega328P microcontroller)
    - SRAM: 2 KB (ATmega328P microcontroller)
    - EEPROM: 1 KB (ATmega328P microcontroller)
    - Clock speed: 16 MHz (ATmega328P microcontroller)
2. Servo Motor (Code: M1)
    - Type: DC motor with a built-in gear and feedback mechanism
    - Operating voltage: 4.8V to 6V
    - Stall torque: 1.8 to 12 kg-cm
    - Rotation angle: 0° to 180°
    - Control signal: Pulse width modulation (PWM)
    - Signal frequency: 50 Hz

MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

Traffic Light Simulation using Raspberry Pi

Assignment 4

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

Consider a suitable scenario of traffic signalling considering a crossroad and demonstrate the working of traffic lights using Raspberry Pi.

2 Objectives

- To simulate a traffic signal using Raspberry Pi.
- To demonstrate the working of traffic lights using Raspberry Pi.

3 Component List

Equipment Name	**Quantity**
Raspberry Pi Model 3 B	1
LEDs - Green, Yellow, Red	3

4 Theory

In this project, the Raspberry Pi will be used to control the traffic lights, which will be represented using LEDs. The project will involve programming the Raspberry Pi to alternate the lights in a manner similar to real-world traffic lights. This will require knowledge of programming languages such as Python, as well as knowledge of the GPIO pins on the Raspberry Pi, which are used to interface with external components.

IoT concepts that may be involved in this project include the use of sensors to detect the presence of vehicles or pedestrians, which could be used to control the timing of the traffic lights. For example, if a sensor detects a vehicle waiting at a red light, it could signal the Raspberry Pi to shorten the duration of the red light and increase the duration of the green light to allow the vehicle to proceed. This concept is known as "smart traffic management" and is an example of how IoT can be used to improve transportation infrastructure.

Additionally, the Raspberry Pi could be connected to a network and communicate with other devices in the IoT ecosystem. For example, it could send data about traffic flow to a central server, which could be used to analyze traffic patterns and make decisions about traffic management. This is an example of how IoT can be used to gather and analyze large amounts of data to improve decision-making processes.

Overall, this project is an example of how IoT can be used to create innovative solutions for real-world problems. By simulating traffic lights using a Raspberry Pi, students can learn valuable programming and electronics skills while also gaining a deeper understanding of IoT concepts such as smart traffic management and data analysis.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Thonny
Compilers : Python 3.10 in Raspberry Pi.

6 Circuit Diagram

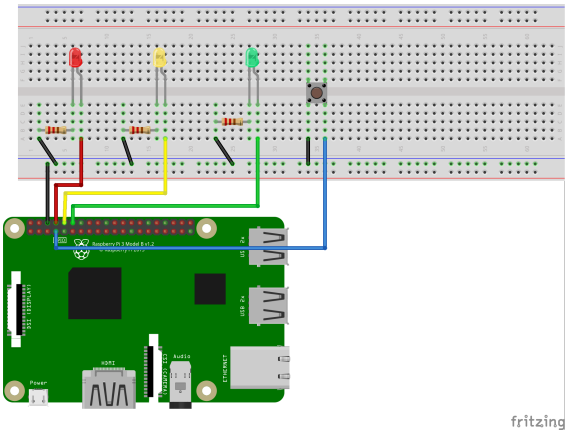

Figure 1: Circuit Diagram for Traffic Signal

7 Code

```
1
2  from gpiozero import LED
3  from gpiozero import Button
4  from gpiozero import TrafficLights
5  import time
6
7  # TrafficLights(red, amber, green)
8  lights = TrafficLights(25, 8, 7)
9  button = Button(21)
10
11
12 def traffic(hour):
13   t = time.localtime()
14   h = t.tm_hour
15   lights.amber.on()
16   time.sleep(2)
17   lights.amber.off()
18   if (21 > h and 6 < h):
19   lights.green.on()
20   time.sleep(5)
21   lights.green.off()
22   else:
23   lights.red.on()
24   time.sleep(5)
25   lights.red.off()
26
27 def wakey(hour, min):
```

```
    cnt = 0
    while cnt < 10:
    if (hour == 6 and min == 40):
      lights.green.on()
      lights.red.on()
      lights.amber.on()
      time.sleep(1)
      lights.green.off()
      lights.red.off()
      lights.amber.off()
      time.sleep(1)
    cnt+=1

  while True:
    w = time.localtime()
    wakey(w.tm_hour, w.tm_min)
    button.when_pressed = traffic
```

8 Conclusion

Thus, we have successfully simulated a traffic signal using Raspberry Pi.

9 FAQ

1. Raspberry Pi Model 3 B:
   The Raspberry Pi has a total of 40 pins, including 26 GPIO (General Purpose Input/Output) pins, 3.3V and 5V power pins, and ground pins. The pinout diagram for the Raspberry Pi 3 B can be found on the official Raspberry Pi website.

   - Pin 1: 3.3V
   - Pin 2: 5V
   - Pin 3: GPIO 2
   - Pin 4: 5V
   - Pin 5: GPIO 3
   - Pin 6: Ground
   - Pin 7: GPIO 4
   - Pin 8: GPIO 14
   - Pin 9: Ground
   - Pin 10: GPIO 15
   - Pin 11: GPIO 17
   - Pin 12: GPIO 18
   - Pin 13: GPIO 27
   - Pin 14: Ground
   - Pin 15: GPIO 22
   - Pin 16: GPIO 23
   - Pin 17: 3.3V
   - Pin 18: GPIO 24
   - Pin 19: GPIO 10
   - Pin 20: Ground
   - Pin 21: GPIO 9
   - Pin 22: GPIO 25
   - Pin 23: GPIO 11
   - Pin 24: GPIO 8
   - Pin 25: Ground
   - Pin 26: GPIO 7
   - Pin 27: ID SD
   - Pin 28: ID SC
   - Pin 29: GPIO 5
   - Pin 30: Ground
   - Pin 31: GPIO 6
   - Pin 32: GPIO 12

- Pin 33: GPIO 13
- Pin 34: Ground
- Pin 35: GPIO 19
- Pin 36: GPIO 16
- Pin 37: GPIO 26
- Pin 38: GPIO 20
- Pin 39: Ground
- Pin 40: GPIO 21

2. LED:

   LEDs are diodes that emit light when an electric current passes through them. They are used as indicator lamps in many devices, and are increasingly used for lighting.

   The LED is based on the semiconductor diode. When a diode is forward biased (switched on), electrons are able to recombine with holes within the device, releasing energy in the form of photons.

   This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor. LEDs are typically small (less than 1 mm2) and integrated optical components may be used to shape the radiation pattern.

   The Pins are:

   - Anode
   - Cathode

MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

Obstacle Detection and Notification using Arduino

Assignment 5

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To simulate an operation of obstacle detection and notifying it with a buzzer or LED using Raspberry-Pi/Beagle bone black board/ Tinker CAD Arduino etc.

2 Objectives

1. To develop an obstacle detection system using Arduino, ultrasound sensor, buzzer, and LED and or DC Motor.
2. To demonstrate when the ultrasonic sensor detects an obstacle at an distance(3cm) the Arduino orders the buzzer to ring and the red LED to light up.

3 Equipments Used

Equipment Name	**Quantity**
Raspberry Pi Model 3 B	1
DC Motor	1
Ultrasonic Sensor	1

4 Theory

The objective of this project is to develop an obstacle detection system using an Arduino microcontroller, an ultrasonic sensor, a buzzer, and a LED or a DC motor. The ultrasonic sensor is used to detect obstacles within a certain range, and when an obstacle is detected, the Arduino will activate the buzzer and the LED or the DC motor. This system can be useful in various applications, such as in automobiles, robots, and security systems, where obstacle detection is important.

In this project, the ultrasonic sensor is used to measure the distance between the sensor and the obstacle. The sensor emits high-frequency sound waves and then listens for the echo of those waves bouncing back from the obstacle. The time between the transmission and reception of the sound waves is used to calculate the distance between the sensor and the obstacle.

The ultrasonic sensor used in this project is the HC-SR04, which has four pins: VCC, GND, Trigger, and Echo. The VCC and GND pins are connected to the 5V and GND pins of the Arduino, respectively. The Trigger pin is used to send the ultrasonic sound wave, and the Echo pin is used to receive the echo of the sound wave.

The LED or the DC motor is used to provide a visual or a physical indication of the obstacle detection. The LED is connected to a digital output pin of the Arduino, and it can be turned on or off by writing a high or low value to that pin.

The DC motor is connected to a motor driver module, which is connected to two digital output pins of the Arduino. By varying the voltage and the polarity of the output pins, the direction and the speed of the DC motor can be controlled.

In the context of the Internet of Things (IoT), this project can be extended to include wireless communication between the Arduino and other devices or systems. For example, the Arduino can be connected to a Wi-Fi or a Bluetooth module, which can be used to send the obstacle detection data to a cloud server or a smartphone application. The data can then be processed and analyzed to provide insights and actions based on the detected obstacles.

This can be useful in smart city applications, where the traffic flow can be optimized based on the real-time obstacle detection data. The use of wireless communication in IoT also enables remote monitoring and control of the obstacle detection system, making it more efficient and convenient.

5 Circuit Diagram

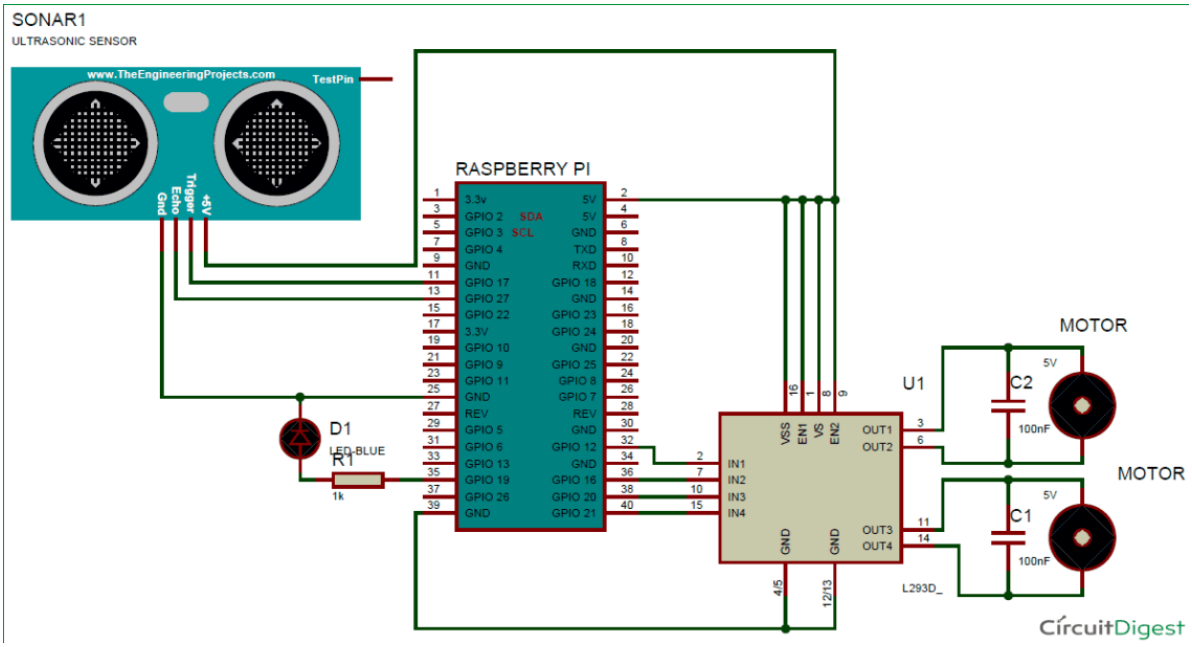

Figure 1: Circuit Diagram

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code and Tinkercad, thonny on Pi

7 Input

Data from the Ultrasonic sensor is given to pi.

8 Output

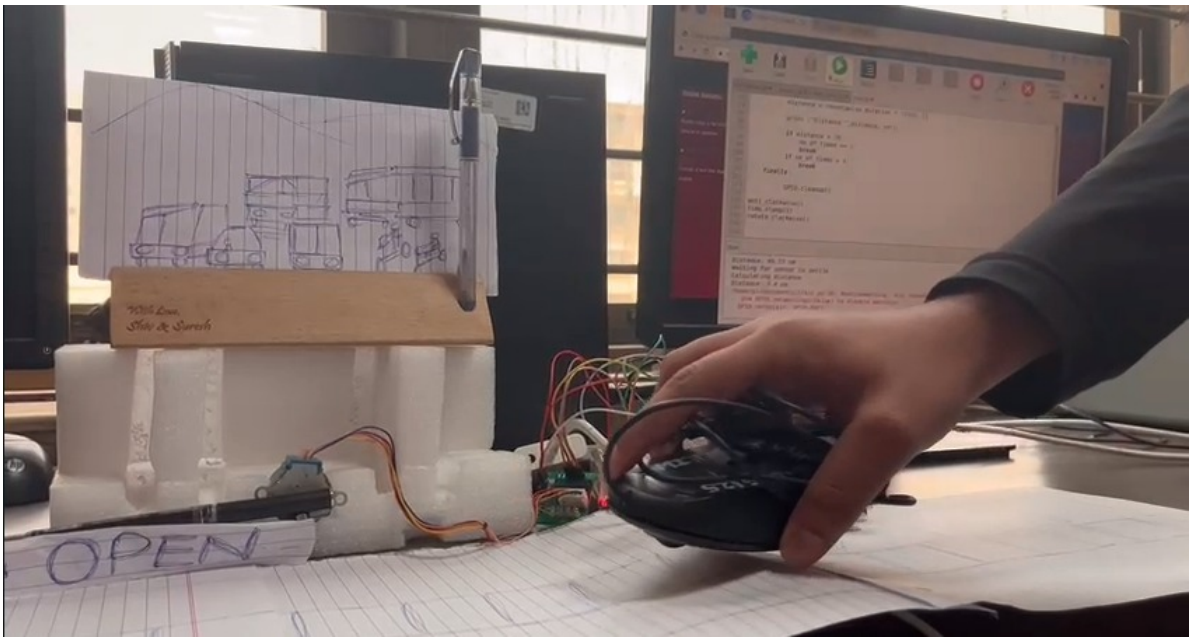

Figure 2: Demonstration of the Project

9 Code

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
control_pins = [7,11,13,15]

#Clockwise direction
for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)
halfstep_seq = [
  [1,0,0,0],
  [1,1,0,0],
  [0,1,0,0],
  [0,1,1,0],
  [0,0,1,0],
  [0,0,1,1],
  [0,0,0,1],
  [1,0,0,1]
]
for i in range(512):
  for halfstep in range(8):
  for pin in range(4):
    GPIO.output(control_pins[pin], halfstep_seq[halfstep][pin])
  time.sleep(0.002)

#Anti Clockwise direction
for pin in control_pins:
```

```
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, 0)
  halfstep_seq = [
    [1,0,0,1],
    [0,0,0,1],
    [0,0,1,1],
    [0,0,1,0],
    [0,1,1,0],
    [0,1,0,0],
    [1,1,0,0],
    [1,0,0,0]
  ]
  for i in range(512):
    for halfstep in range(8):
    for pin in range(4):
      GPIO.output(control_pins[pin], halfstep_seq[halfstep][pin])
    time.sleep(0.002)
  GPIO.cleanup()

  print ("Script for Controlling Distance Sensor")

try:
      GPIO.setmode(GPIO.BOARD)
      PIN_TRIGGER = 7
      PIN_ECHO = 11
      GPIO.setup(PIN_TRIGGER, GPIO.OUT)
      GPIO.setup(PIN_ECHO, GPIO.IN)
      GPIO.output(PIN_TRIGGER, GPIO.LOW)
      print ("Waiting for sensor to settle")
      time.sleep(2)
      print ("Calculating distance")
      GPIO.output(PIN_TRIGGER, GPIO.HIGH)
      time.sleep(0.00001)
      GPIO.output(PIN_TRIGGER, GPIO.LOW)
      while GPIO.input(PIN_ECHO)==0:
            pulse_start_time = time.time()
      while GPIO.input(PIN_ECHO)==1:
            pulse_end_time = time.time()
      pulse_duration = pulse_end_time - pulse_start_time
      distance = round(pulse_duration * 17150, 2)
      print ("Distance:",distance,"cm")

finally:

      GPIO.cleanup()
```

10 Conclusion

Thus, we have successfully simulated an operation of obstacle detection and notifying it with a DC motor.

11 FAQ

Details of the main components used in this project and their pin specifications:

1. Raspberry Pi Model 3 B:
   The Raspberry Pi has a total of 40 pins, including 26 GPIO (General Purpose Input/Output) pins, 3.3V and 5V power pins, and ground pins. The pinout diagram for the Raspberry Pi 3 B can be found on the official Raspberry Pi website.

   - Pin 1: 3.3V
   - Pin 2: 5V
   - Pin 3: GPIO 2
   - Pin 4: 5V
   - Pin 5: GPIO 3
   - Pin 6: Ground
   - Pin 7: GPIO 4
   - Pin 8: GPIO 14
   - Pin 9: Ground
   - Pin 10: GPIO 15
   - Pin 11: GPIO 17
   - Pin 12: GPIO 18
   - Pin 13: GPIO 27
   - Pin 14: Ground
   - Pin 15: GPIO 22
   - Pin 16: GPIO 23
   - Pin 17: 3.3V
   - Pin 18: GPIO 24
   - Pin 19: GPIO 10
   - Pin 20: Ground
   - Pin 21: GPIO 9
   - Pin 22: GPIO 25
   - Pin 23: GPIO 11
   - Pin 24: GPIO 8
   - Pin 25: Ground
   - Pin 26: GPIO 7
   - Pin 27: ID SD
   - Pin 28: ID SC
   - Pin 29: GPIO 5
   - Pin 30: Ground
   - Pin 31: GPIO 6

- Pin 32: GPIO 12
- Pin 33: GPIO 13
- Pin 34: Ground
- Pin 35: GPIO 19
- Pin 36: GPIO 16
- Pin 37: GPIO 26
- Pin 38: GPIO 20
- Pin 39: Ground
- Pin 40: GPIO 21

2. DC Motor:
   A DC motor typically has two wires for power and two wires for control. The power wires are typically red and black, and the control wires can be any other color. The control wires are used to vary the voltage and polarity of the applied power to control the speed and direction of the motor.

3. Ultrasonic Sensor (HC-SR04): The HC-SR04 ultrasonic sensor has four pins: Vcc, Trig, Echo, and Gnd. The pin specifications are as follows:

   - Vcc: This pin is used to provide power to the sensor. It typically requires a voltage of 5V, but it can also be powered using 3.3V.
   - Trig: This pin is a digital output pin that is used to trigger the sensor. It sends a 10µs pulse to the sensor to start the measurement.
   - Echo: This pin is a digital input pin that receives the echo signal. It measures the time taken for the ultrasonic waves to travel to the object and back, and sends a pulse back to the Raspberry Pi to indicate the distance.
   - Gnd: This pin is used to connect the sensor to ground.

MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

MQTT Protocol Implementation using Tinkercad on Arduino UNO

Assignment 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To understand the working of MQTT Protocol and implement it using Tinkercad on Arduino UNO.

2 Objectives

1. To understand the working of MQTT Protocol.
2. To use Arduino UNO to implement the MQTT Protocol.
3. To sense data from sensors and send it to cloud.

3 Equipments Used

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	Temperature Sensor [TMP36]
U3	1	Wifi Module (ESP8266)
R1 R2	2	1 kΩ Resistor

4 Theory

Our project involves sensing the temperature using the TMP36 sensor and sending the data to the cloud using the ESP8266 module and MQTT protocol. This is an example of **sensing**.

4.1 MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that is widely used in IoT (Internet of Things) applications for sending and receiving data between devices and the cloud. It uses a publish/subscribe messaging model, where devices (publishers) send messages to a central server (broker), which then distributes the messages to other devices (subscribers) that have subscribed to the topic of the message.

4.2 Connection

In our project, we use the ESP8266 module to connect to the Internet and publish the temperature data to a cloud platform called Thingspeak using the MQTT protocol. Thingspeak is a platform that allows users to collect, analyze, and visualize data from IoT devices. It provides an MQTT broker that can be used to publish and subscribe to data streams, as well as a web interface for displaying and analyzing the data in real-time.

To use MQTT in our project, we would need to first establish a connection to the broker using the ESP8266 module and then publish the temperature data to a specific topic on the broker. We could then use the Thingspeak API to retrieve the data from the broker and display it in a chart or graph on the Thingspeak web interface.

5 Circuit Diagram

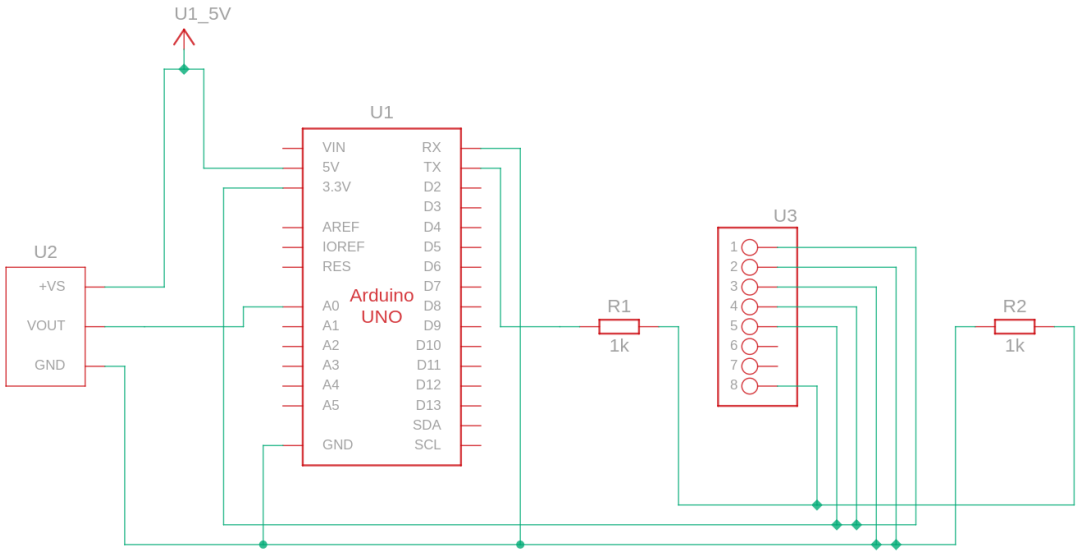

Figure 1: Circuit Diagram

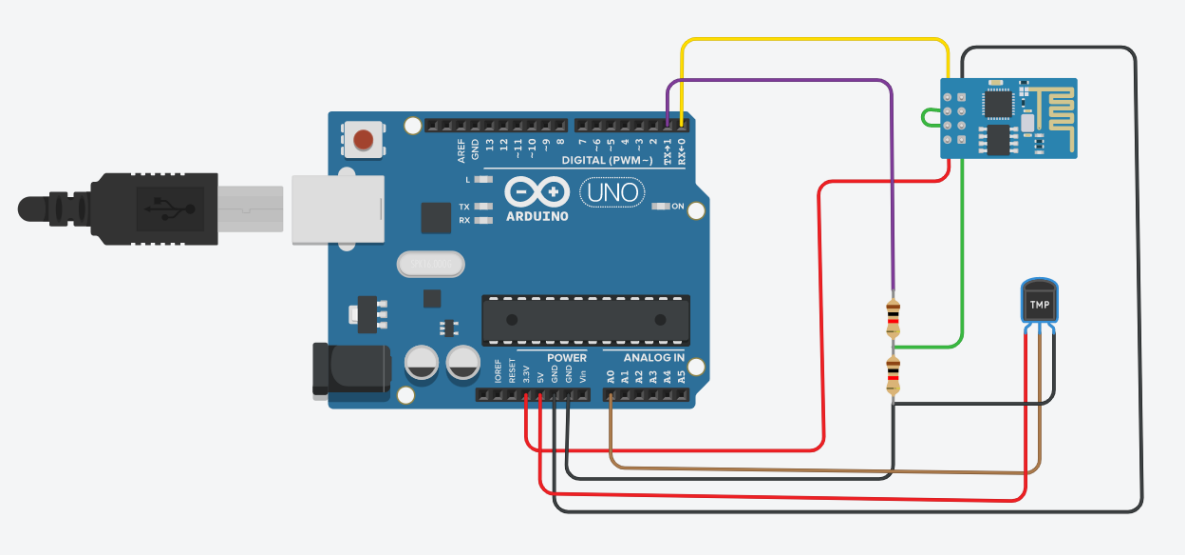

Figure 2: Circuit

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code and Tinkercad

7 Input

Data from the Temperature Sensor is sent to the cloud using the MQTT Protocol.

8 Output

```
CIPSEND=89
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=76.47 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=89
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=76.47 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=199.52 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=256.65 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=108.99 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV2O1YIK6KJ2SC &field1=-40.42 HTTP/1.1
Host: api.thingspeak.com
```

9 Code

```
float val, voltage, temp,tempf;
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = ""; //virtual wifi has no password
String host     = "api.thingspeak.com"; // Open Weather Map API
const int httpPort   = 80;
String url     = "/update?api_key=SGXV2O1YIK6KJ2SC";
String url1     = " &field1=";
//Replace XXXXXXXXXXXXXXXX by your ThingSpeak Channel API Key

void setupESP8266(void) {

  // Start our ESP8266 Serial Communication
  Serial.begin(115200);   // Serial connection over USB to computer
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("ESP8266 OK!!!");

```

```
  // Connect to Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");



  delay(10);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("Connected to WiFi!!!");

  // Open TCP connection to the host:
  //ESP8266 connects to the server as a TCP client.

  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("ESP8266 Connected to server!!!") ;

}

void anydata(void) {

  val=analogRead(A0);
  voltage=val*0.0048828125;
  temp = (voltage - 0.5) * 100.0;
  tempf=(temp*9/5)+32;
  //distance=12;

  // Construct our HTTP call
  String httpPacket = "GET " + url  + url1 + String(tempf) + " HTTP/1.1\r\nHost: "
     + host + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return
     -1;

  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (Serial.find("SEND OK\r\n"))
  Serial.println("ESP8266 sends data to the server");

}

void setup() {
  pinMode(A0, INPUT);
  setupESP8266();

}

void loop() {

  anydata();

  delay(100);
}
```

10 Conclusion

Thus, we have successfully sent data from the Temperature Sensor to the cloud using the MQTT Protocol.

11 FAQ

Details of the main components used in this project and their pin specifications:

1. Arduino Uno R3 (Code: U1)
    - Type: Microcontroller board
    - Digital pins: 14
    - Analog input pins: 6
    - Operating voltage: 5V
    - Input voltage (recommended): 7-12V
    - Input voltage (limits): 6-20V
    - Flash memory: 32 KB (ATmega328P microcontroller)
    - SRAM: 2 KB (ATmega328P microcontroller)
    - EEPROM: 1 KB (ATmega328P microcontroller)
    - Clock speed: 16 MHz (ATmega328P microcontroller)
2. Temperature Sensor [TMP36] (Code: U2)
    - Type: Analog sensor
    - Pin 1: Output voltage (analog)
    - Pin 2: Ground
    - Pin 3: Input voltage (3.3V - 5V)
    - Temperature range: -40°C to +125°C
    - Output voltage range: 0V to 1.75V (at 25°C)
3. Wifi Module [ESP8266] (Code: U3)
    - Type: Digital communication module
    - Pin 1: VCC (3.3V)
    - Pin 2: Ground
    - Pin 3: TXD (Transmit Data)
    - Pin 4: RXD (Receive Data)
    - Operating voltage: 3.3V
    - Communication protocol: UART
    - Wireless standard: IEEE 802.11 b/g/n
4. 1 kΩ Resistor (Codes: R1 and R2)
    - Type: Resistor
    - Resistance: 1 kΩ (1000 ohms)
    - Tolerance: 5%
    - Power rating: 1/4 watt
    - Quantity: 2