Cover sheet for submission of work for assessment

SWINBURNE UNIVERSITY OF TECHNOLOGY

UNIT DETAILS

Unit name	IOT Programming			Class day/time	Friday 12:30	Office use only
Unit code	SWE3001	Assignment no.	3	Due date	01/06/2024	
Name of lecturer/teacher	Armita					
Tutor/marker's name						Faculty or school date stamp

STUDENT(S)

	Family Name(s)	Given Name(s)	Student ID Number(s)
(1)	Andy	Ibrahim	100685973
(2)			
(3)			
(4)			
(5)			
(6)			

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Andy Ibrahim	(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/**

Table of Contents

1. Introduction

1.1 Background

The Internet of Things has emerged as a transformative technology, reshaping industries and enabling connectivity and automation. By integrating physical devices with digital systems, IoT provided new possibilities for monitoring control and optimization across various domains.

In the context of environmental monitoring, IoT offers significant potential for improving resource management, enhancing safety, and reducing environmental impact. Traditional methods of environmental data collection often involve manual Labor and are limited in scope and accuracy. IoT systems, on the other hand provide real time data that can be analysed and acted upon instantly.

1.2 Proposed System

Our project aims to leverage IoT technology to create a comprehensive environmental monitoring system. The proposed system integrates Arduino microcontrollers, DHT11 temperature and humidity sensors, LED's and a buzzer to monitor and control environmental conditions in real time.

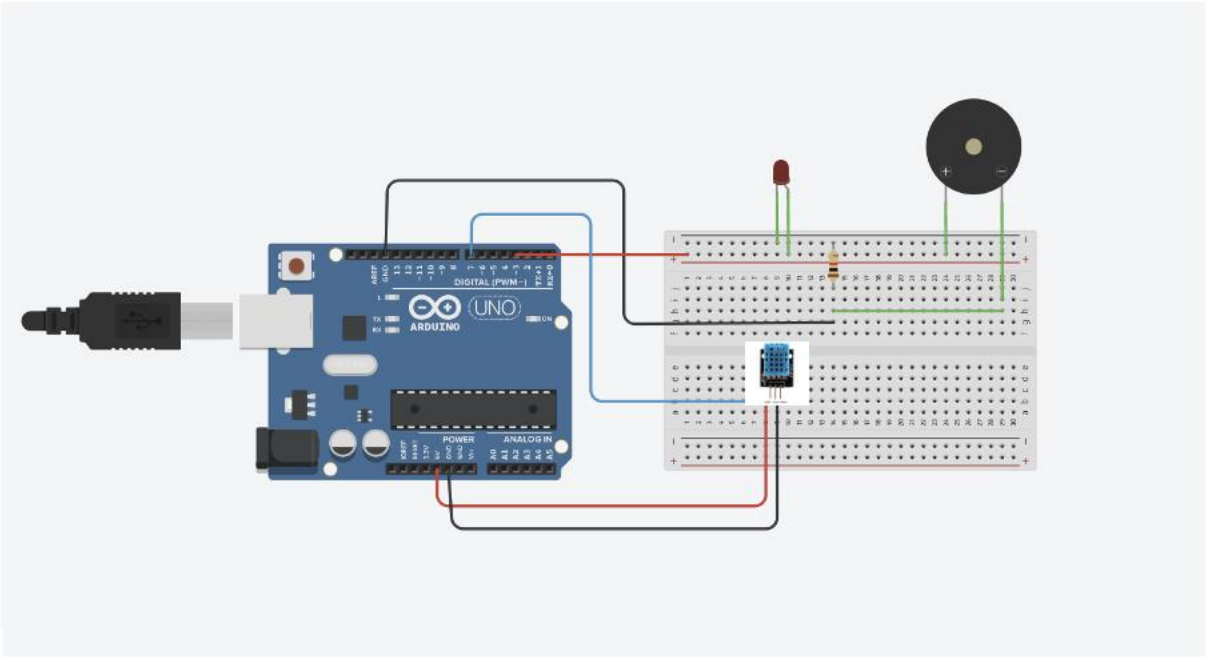
The DHT11 sensors are utilized to capture temperature and humidity data from the surrounding environment. These sensors are cost effective and offer reasonable accuracy, making them suitable for this application.

The Arduino microcontrollers serve as the brain of the system, processing sensor data and initiating actions based on predefined thresholds. LED's and a buzzer are employed as actuators to provide visual and auditory feedback in response to environmental conditions.

Data collected by the sensors are transmitted to a cloud platform, ThingsBoard, using MQTT protocol. ThingsBoard provides a scalable and flexible solution for data storage, visualization and analysis, enabling users to monitor environmental conditions remotely and in real time.

The system is programmed to take predefined actions when environmental conditions exceed or fall below certain threshold. For example, if the temperature surpasses a specified limit, the system may activate the LEDs to indicate overheating or trigger the buzzer to raise an alarm.
By combining sensor data acquisition, cloud-based data processing, and actuator control, the IoT system offers a comprehensive solution for environmental monitoring and management.

2. Conceptual design



3. Tasks Breakdown

Task	Description
Hardware Set Up	Set up hardware components including sensors, Arduino, LEDs & Buzzer
Sensor Integration	Connect DHT11 temperature and humidity sensors to the Arduino
Actuator Integration	Connect LEDs and buzzer to the Arduino for threshold-based control
Circuit Assembly	Assemble the circuit including wiring and component placement.
Physical Testing	Test hardware connections, ensure sensors and actuators are functioning properly.
Troubleshooting	Debug issues related to hardware connections and component functionality

4. Implementation

4.1 Sensing System

The DHT11 temperature and humidity sensor is connected to the Arduino microcontroller to collect environmental data.

4.2 Edge Servers

The Arduino serves as the edge device in our system, responsible for data processing and actuator control.

4.3 Communication Protocols

Serial communication is established between the Arduino and the computer for data transmission. MQTT protocol is used for communication between the computer and the ThingsBoard cloud platform.

4.4 API or Website Details

The ThingsBoard cloud platform provides APIs for data retrieval and visualization. Users can access the data and control the system through the ThingsBoard dashboard.

4.5 Cloud Computing

ThingsBoard cloud platform hosts the backend infrastructure for data storage, processing, and visualization.

5. User Manual

5.1 Operation

1. Hardware Setup

1. Connect the DHT11 temperature and humidity sensors to the Arduino microcontroller according to the provided schematic or pinout diagram.
2. Connect the LEDs and buzzer to the appropriate digital pins on the Arduino for actuator control.
3. Ensure all connections are secure and correctly wired to avoid any potential issues during operation.

2. Software Setup

1. Upload the provided Arduino sketch to the microcontroller using the Arduino IDE or any compatible programming software.

2. Make sure to select the correct board and port settings in the Arduino IDE before uploading the sketch.

3. MQTT Communication Setup

1. Run the provided Python script on the edge server or computer connected to the Arduino.
2. Modify the Python script if necessary to specify the correct serial port for communication with the Arduino.
3. Ensure that the Python environment has the required libraries installed, such as paho-mqtt for MQTT communication.

4. ThingsBoard Integration

1. Access the ThingsBoard dashboard through a web browser using the provided login credentials.
2. Set up a new device in ThingsBoard to receive data from the Arduino and visualize environmental conditions.
3. Obtain the device access token from ThingsBoard and update the Python script with this token for authentication.

5. Monitoring and Control

1. Once the hardware and software setup are complete, monitor environmental data such as temperature and humidity in real-time through the ThingsBoard dashboard.
2. Set up a new device in ThingsBoard to receive data from the Arduino and visualize environmental conditions.
3. Use the dashboard controls to interact with the system, such as adjusting threshold values, activating/deactivating actuators, or performing remote diagnostics.

Limitations

The IoT system has limitations that may impact its performance and reliability. Firstly, its sensor data transmission range is limited due to serial communication, restricting its deployment in large-scale or remote environments. Additionally, reliability issues may arise from environmental factors affecting the serial communication between the Arduino and computer, potentially leading to data loss or connection issues. Moreover, the system is dependent on internet connectivity for cloud-based data storage and visualization, making it vulnerable to disruptions in internet

access or service outages. Furthermore, a stable power supply is essential for continuous system operation, and power fluctuations or interruptions can impact data collection and functionality. Lastly, there are security concerns regarding unauthorized access or data breaches, particularly when relying on third-party cloud services for data storage and management.

7. Resources

7.1 Arduino Documentation and Tutorials

We relied on the official Arduino documentation and online tutorials to understand how to use Arduino boards, write code, and connect hardware components.

7.2 DHT Library for DHT11 Sensor

We used a library specifically designed for the DHT11 sensor, which made it easier to read temperature and humidity data accurately.

7.3 Paho MQTT Library for Python

To communicate with the ThingsBoard cloud platform, we used the Paho MQTT library in Python. It simplified the process of sending and receiving data over the MQTT protocol.

7.4 ThingsBoard Documentation for MQTT Integration

The documentation provided by ThingsBoard helped us set up our IoT devices to communicate with the ThingsBoard cloud platform using MQTT.

```ino
#include <DHT.h>

#define DHTPIN 7
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

#define LED_PIN 3
#define BUZZER_PIN 4

float tempThreshold = 30.0;
float humidityThreshold = 50.0;

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

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

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  Serial.print(temperature);
  Serial.print(",");
  Serial.print(humidity);
  Serial.println();

  if (temperature > tempThreshold && humidity > humidityThreshold) {

    for (int i = 0; i < 10; i++) {
        digitalWrite(LED_PIN, HIGH);
        digitalWrite(BUZZER_PIN, HIGH);
        delay(500);
        digitalWrite(LED_PIN, LOW);
        digitalWrite(BUZZER_PIN, LOW);
        delay(500);
    }
    exit(0);
  } else if (temperature > tempThreshold) {

    for (int i = 0; i < 10; i++) {
```

```
    exit(0);
  } else if (temperature > tempThreshold) {

    for (int i = 0; i < 10; i++) {
        digitalWrite(LED_PIN, HIGH);
        delay(500);
        digitalWrite(LED_PIN, LOW);
        delay(500);
    }
    exit(0);
  } else if (humidity > humidityThreshold) {

    for (int i = 0; i < 10; i++) {
        digitalWrite(BUZZER_PIN, HIGH);
        delay(500);
        digitalWrite(BUZZER_PIN, LOW);
        delay(500);
    exit(0);
    }

  } else {

    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
  }

  delay(2000);
}

```

```
import time

ser = serial.Serial('/dev/tty.usbmodem11201', 9600)
print(ser)

def on_publish(client, userdata, result):
    print("Data published to ThingsBoard")

def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker")

client1 = paho.Client(client_id="tOHcQXIUUUcOyrF6FFti")
client1.on_publish = on_publish
client1.on_connect = on_connect
client1.username_pw_set('tOHcQXIUUUcOyrF6FFti')
client1.connect("thingsboard.cloud", 1883, keepalive=60)

tempThreshold = 30.0
humidityThreshold = 50.0

while True:
    try:
        data = ser.readline().decode().rstrip()
        temperature_str, humidity_str = data.split(',')
        temperature = float(temperature_str)
        humidity = float(humidity_str)

        payload = '{{"Temperature": {0}, "Humidity": {1}}}'.format(temperature, humidity)
        client1.publish("v1/devices/me/telemetry", payload)
        print("Device telemetry updated")
        print(payload)

        if temperature > tempThreshold:
            ser.write(b'LED_ON\n')
        else:
            ser.write(b'LED_OFF\n')

        if humidity > humidityThreshold:
            ser.write(b'BUZZER_ON\n')
        else:
            ser.write(b'BUZZER_OFF\n')

    except ValueError as e:
        print(f"Error parsing data: {e}")
    except serial.SerialException as e:
        print(f"Serial error: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")

    time.sleep(5)
```