UNIVERSITI MALAYSIA SARAWAK
Faculty of Computer Science and Information Technology

Assignment Title	Group Project	
Subject Code	TMN4033	**Subject Name:** Embedded System

Name	**Matric No**	**E-mail**	**Signature**
CHRISTOPHER SII HOW CHIONG	69385	69385@siswa.unimas.my	Chris
HO WAN YU	69860	69860@siswa.unimas.my	Wan Yu
SONG WANG YE	71680	71680@siswa.unimas.my	Swye

Name of Lecturer: Associate Professor Dr. Kartinah Bt Zen	
Due Date: 2nd January 2023	Date received and approved (for office use only)

Plagiarism and Collusion are methods of cheating that falls under PeraturanAkademikUniversiti Malaysia Sarawak para 11: EtikaAkademik

Plagiarism

Plagiarism is the presentation of work which has been copied in whole or in part from another person's work, or from any other source such as the internet, published books or periodicals without due acknowledgement given in the text.

Collusion

Collusion is the presentation of work that is the result in whole or in part of unauthorized collaboration with another person or persons.

Where there are reasonable grounds for believing that cheating has occurred, the only action that may be taken when plagiarism or collusion is detected is for the staff member not to mark the item of work and to report or refer the matter to the Dean. This may result in work being disallowed and given a fail grade or if the circumstances warrant, the matter may be referred to a Committee of inquiry for investigation. Such investigation may result in the matter being referred to the University Discipline Committee, which has the power to exclude a student.

Note: Assignments are returned after being assessed by lecturer and to be collected from the lecturer's office. Students should endeavour to collect their assignments prior to commencement of next semester. All unclaimed assignments become the property of the faculty and will become subject to destruction.

Student's statement:

I certify that I have not plagiarized the work of others or participated in unauthorized collusion when preparing this assignment. I also certify that I have taken proper case in safeguarding my work and have made all reasonable efforts to ensure that may work not be able to be copied.

MARK :	Comments:

Table of Contents

List of Figure

List of Table

Step 2.0 Add/ Edit part

2.1 Add/ Edit function

The subsequent lists contain additional and revised functionality:

- Add a melody function for the buzzer to alert when there is a critical flood condition.
- Add libraries and functions for connection to mobile Arduino IoT Cloud
- Change threshold distance value

The code explanation of these additional and revised changes are described in section 2.3.

2.2 Explanation on hardware

The following are components required for flood monitoring system:

Table 1: Hardware components

Hardware	Description
ESP8266 NodeMCU	Used to connect sensors and let sensor's data transfer using the Wi-Fi protocol.
Ultrasonic Sensor (HC-SR04)	Used to measure the water level then transmits the data to the NodeMCU.
LEDs (Red & Green)	Act as indicator. Red LED is used to signal critical flood conditions, while a green LED indicates normal condition.
Breadboard	Used for creating electrical connections between sensor components and ESP8266 NodeMCU microcontroller
Jumpers	Used to pass power and data utilising GPIO ports by connecting devices on the breadboard.
Buzzer	Play the sound as alert when there is a critical flood condition.

Figures 4 and 5 show the hardware and circuit diagram of the proposed flood monitoring system and an additional buzzer, respectively. The addition buzzer's positive pin is connected to the ESP8266's D5 pin, while the negative pin is connected to the ESP8266's GND pin. Besides,

the ultrasonic sensor's VCC pin is connected to the ESP8266's VIN pin. The sensor's TRIG pin is connected to ESP8266's D1. Moreover, the Echo pin of the ultrasonic sensor is connected to ESP8266's D2 pin. The GND pin of the ultrasonic sensor is connected to ESP8266's GND pin. Furthermore, the cathode pin of the red LED is connected to GND while its anode's pin is connected to ESP8266's D3 pin. The green LED cathode's pin is connected to the GND pin while its anode is connected to ESP8266's D4 pin. During the simulation, if the ultrasonic sensor detects water level is in critical range, the red led will light up and the buzzer will ring. Otherwise, the green led will light up if the ultrasonic sensor indicates safe water level. Figure 5 shows the circuit diagram (the model of Arduino being used in Tinkercad is different, which is Arduino Uno R3 because Tinkercad does not have the ESP8266 model, but its functionality is still the same, therefore the looks may be different).

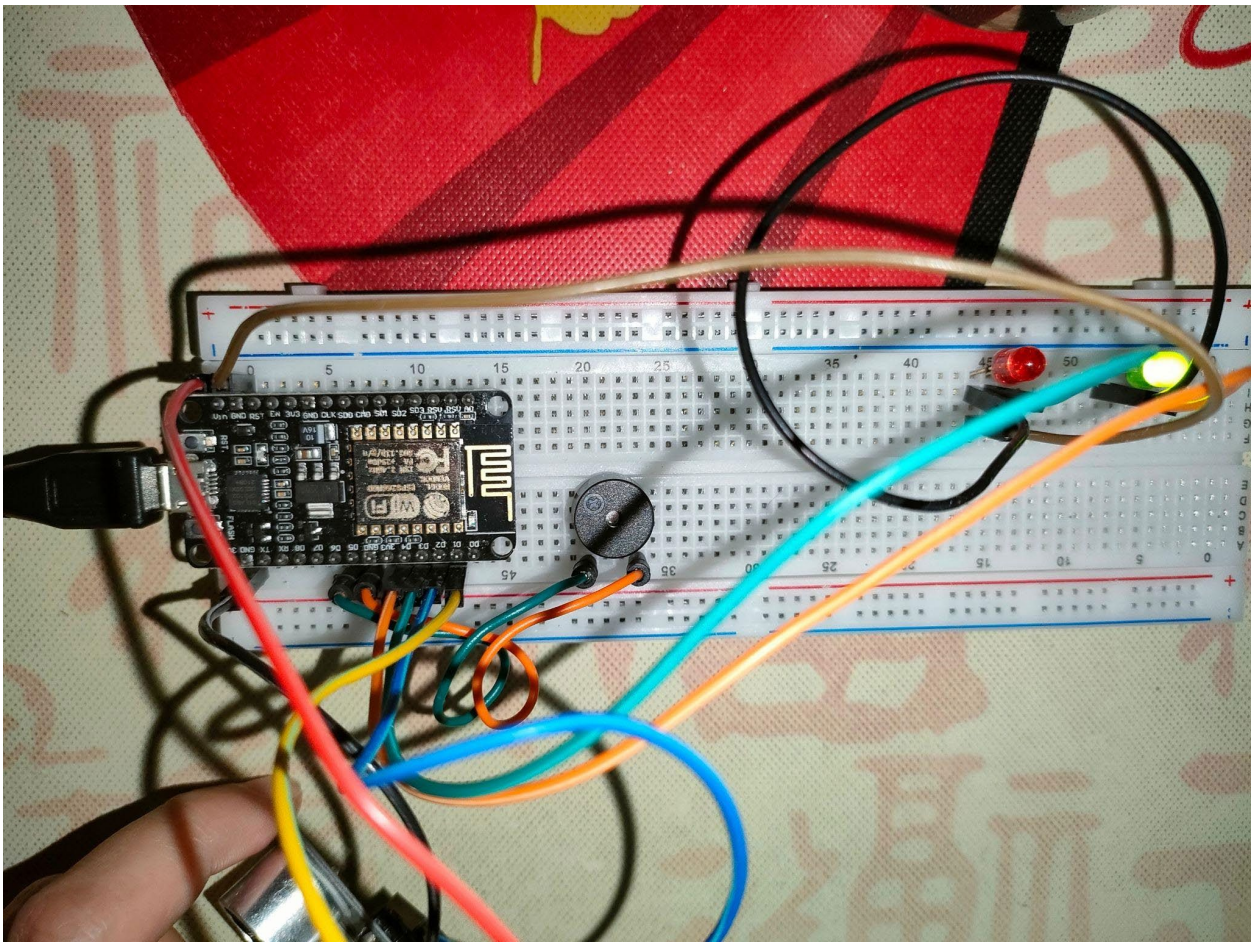

Figure 1 Hardware of the proposed flood monitoring system and an additional buzzer

2.3 Explanation on additional coding

Start the code by including all the new required library files in the code like "thingProperties.h" for ESP8266 board and IoT Cloud Platform, “pitches.h” for buzzer etc.

```
#include "pitches.h"
#include "thingProperties.h"
```

The new additions are the buzzer-based melody123 functionalities. The buzzer will sound when distance 1 is less than or equal to 3 cm. Here, tempo and buzzer variables are created for note in the melody and pin we used respectively. Melody[] array stored note of buzzer. Wholenote is declared to calculate the duration of a whole note in ms. The added new functions are as follows:

```
int tempo = 200;
int buzzer = D5;
int melody[] = { NOTE_E5,8, NOTE_E5,8, REST,8, NOTE_E5,8, REST,8, NOTE_C5,8, NOTE_E5,8, NOTE_G5,4, REST,4, NOTE_G4,8, REST,4, };
int notes = sizeof(melody) / sizeof(melody[0]) / 2;
int wholenote = (60000 * 4) / tempo;
int divider = 0, noteDuration = 0;
```

The new melody123 function below iterates over the note of the melody. It plays sounds following the notes.

```
void melody123(){
        for (int thisNote=0; thisNote<notes*2; thisNote = thisNote + 2){
                divider = melody[thisNote + 1];
                if (divider > 0) {
                noteDuration = (wholenote) / divider;
                } else if (divider < 0) {
                noteDuration = (wholenote) / abs(divider);
                noteDuration *= 1.5;
                }
```

```
        tone(buzzer, melody[thisNote], noteDuration * 0.9);
        delay(noteDuration);
        noTone(buzzer);
}
```

In the setup function, the new pinMode (D5, OUTPUT) is used to configure a specific pin to behave as an output of the new hardware buzzer. In the loop function, the threshold value is changed from the default 10 cm to 3 cm, depending on the study requirement. The melody123 function mentioned above is defined in onRledChange function.

```
void setup() {
        pinMode(D5, OUTPUT);
}
void loop() {
        if (distance1 <= 3){
                onRledChange();

        }else{
                onGledChange();
        }
        void onRledChange()  {
                rled= true;
                gled= false;
                digitalWrite(D3, HIGH);
                digitalWrite(D4, LOW);
                melody();
        }
        void onGledChange()  {
                rled= false;
                gled= true;
```

```
                digitalWrite(D4, HIGH);
                digitalWrite(D3, LOW);
        }
}
```

The following codes are new lines added for connecting to the Arduino IoT Cloud. InitProperties is a function invoked from thingProperties.h library. ArduinoCloud.begin function is used to connect to Arduino IoT cloud. The functions setDebugMessageLevel and printDebugInfo allow obtaining of information related to the state of network and IoT Cloud connection and errors. The higher the number the more granular information. The function update in loop function is to to update the values on the cloud.

```
void setup() {
        // Defined in thingProperties.h
        initProperties();
        // Connect to Arduino IoT Cloud
        ArduinoCloud.begin(ArduinoIoTPreferredConnection);
        setDebugMessageLevel(2);
        ArduinoCloud.printDebugInfo();
}
void loop() {
        ArduinoCloud.update();
}
```

2.4 The project is working

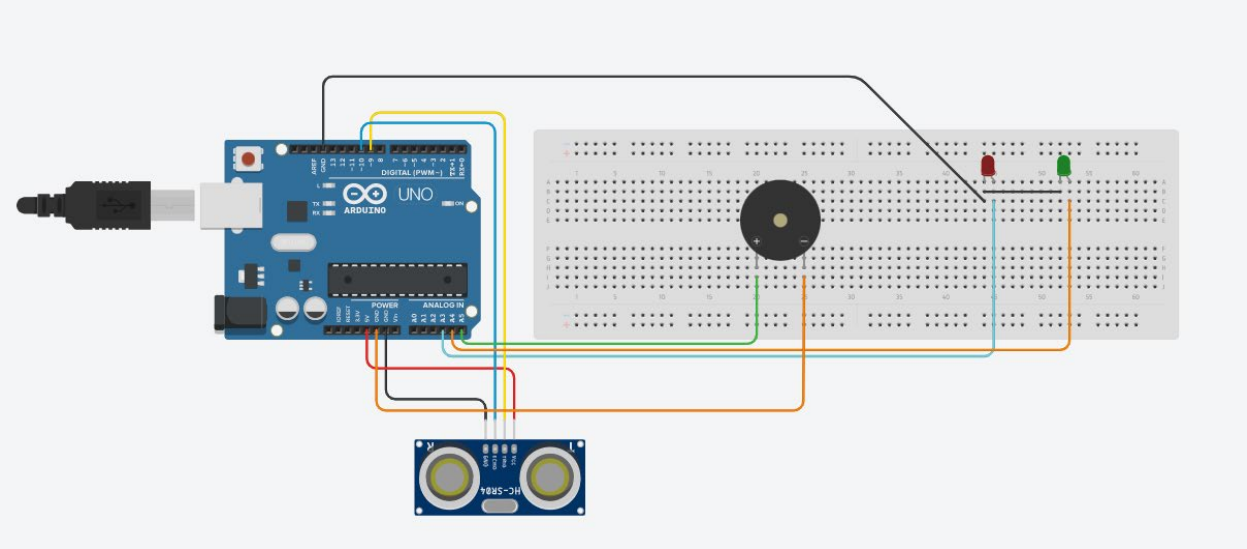

Figure 2 Circuit diagram for the flood monitoring system

YouTube link: https://youtu.be/BR8C6UsVVYo

Step 3.0 Using IoT

3.1 Procedure

Sketch link: https://create.arduino.cc/editor/saflento/eb0b5521-cc43-4cf0-a3d6-d58835ca59ab/preview

Start the code by including all the required library files in the code like "thingProperties.h" for ESP8266 board and IoT Cloud Platform, “pitches.h” for buzzer etc. Here is the step to create the sketch:

1) Click on the create button in IOT Cloud.
2) Select device, need to create new device if no device available.
3) Copy the device’s Secret Key.
4) Configure network, write wifi’s name and password, paste the device’s secret key.
5) Switch to the sketch tab, write the code.
6) Save and upload the sketch and connect to the device.

```
#include "pitches.h"
#include "thingProperties.h"
```

Next, define the pins which are used for the ultrasonic sensors and LEDs.

```
const int trigPin1 = D1;
const int echoPin1 = D2;
#define redled D3
#define grnled D4
```

Now, the network credentials- i.e. SSID and password are defined, which are required to connect the NodeMCU with the internet. Then the IoT Cloud account credentials such as the device key are defined, which were recorded earlier. Make sure to edit the credentials in place of these variables.

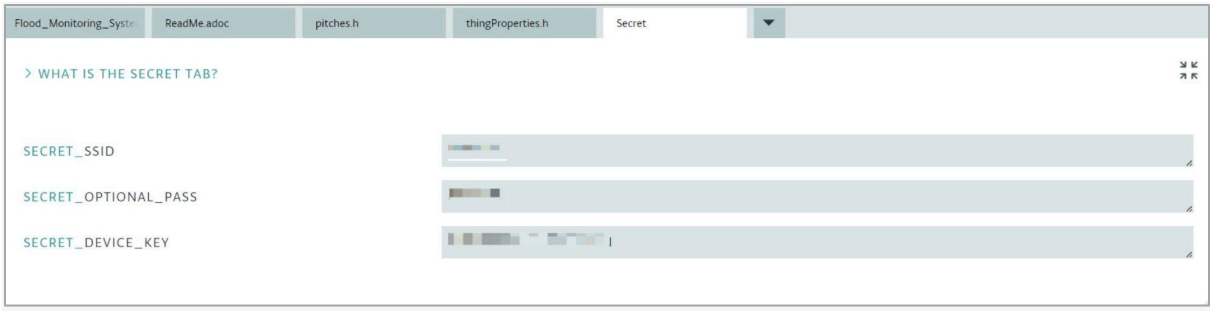

Figure 3 Secret

Then, the variables for timing purposes are defined.

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 10000;
```

Then, to connect NodeMCU to the internet, call **ArduinoCloud.begin** and pass network SSID and password as its arguments. The serial monitor of IOT Cloud will check for the successful network connection using ***WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS)*** and after a successful connection, print a message on Serial Monitor with the SSID. Then connect to the IoT Cloud platform using saved credentials. For this, *ArduinoCloud.begin* is used.

```
ArduinoCloud.begin(ArduinoIoTPreferredConnection);
```

For calculating the distance, an input pulse is given to the sensor through the trig pin of the Ultrasonic sensor. Here as per the HC-SR04 datasheet, a 2-microsecond pulse is given, then from the echo pin, the output pulse of the sensor is read and the distance is calculated in centimeters.

```
digitalWrite(trigPin1, LOW);
delayMicroseconds(2);
digitalWrite(trigPin1, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin1, LOW);
onDistance1Change();
```

Then, an if-else condition is written for the LED indications at both Normal condition and Flood conditions. Here set 3cm as reference.

```
if (distance1 <= 3){
        onRledChange();
 }
 else
 {
        onGledChange();
 }
```

Finally, the value to the river level is uploaded to the IoT Cloud in each 10 seconds interval.

```
 if (currentMillis - startMillis >= period){
        startMillis = currentMillis;
 }
```

IoT Cloud will show a red circle in case of Flood, a green circle if no flood as shown in Figure 5.

3.2 The components

Figure 4 shows the overall hardware components for the flood monitoring system. The components include ESP8266 NodeMCU, Ultrasonic Sensor HC-SR04, LEDs (Red & Green), Breadboard, Jumpers, and Buzzer. ESP8266 NodeMCU is used to connect sensors and let sensor's data transfer using the Wi-Fi protocol. Ultrasonic Sensor HC-SR04 is used to measure the water level then transmits the data to the NodeMCU. Red and green LEDs are used as an indicator to signal critical flood conditions. The red LED lights up when the ultrasonic sensor detects the water level is in critical range while the green LED indicates normal condition. Breadboard is used for creating electrical connections between sensor components and ESP8266 NodeMCU microcontroller. Jumpers are used to pass power and data utilising GPIO ports by connecting devices on the breadboard. Last but not least, the buzzer is used to play the sound as alert when there is a critical flood condition.

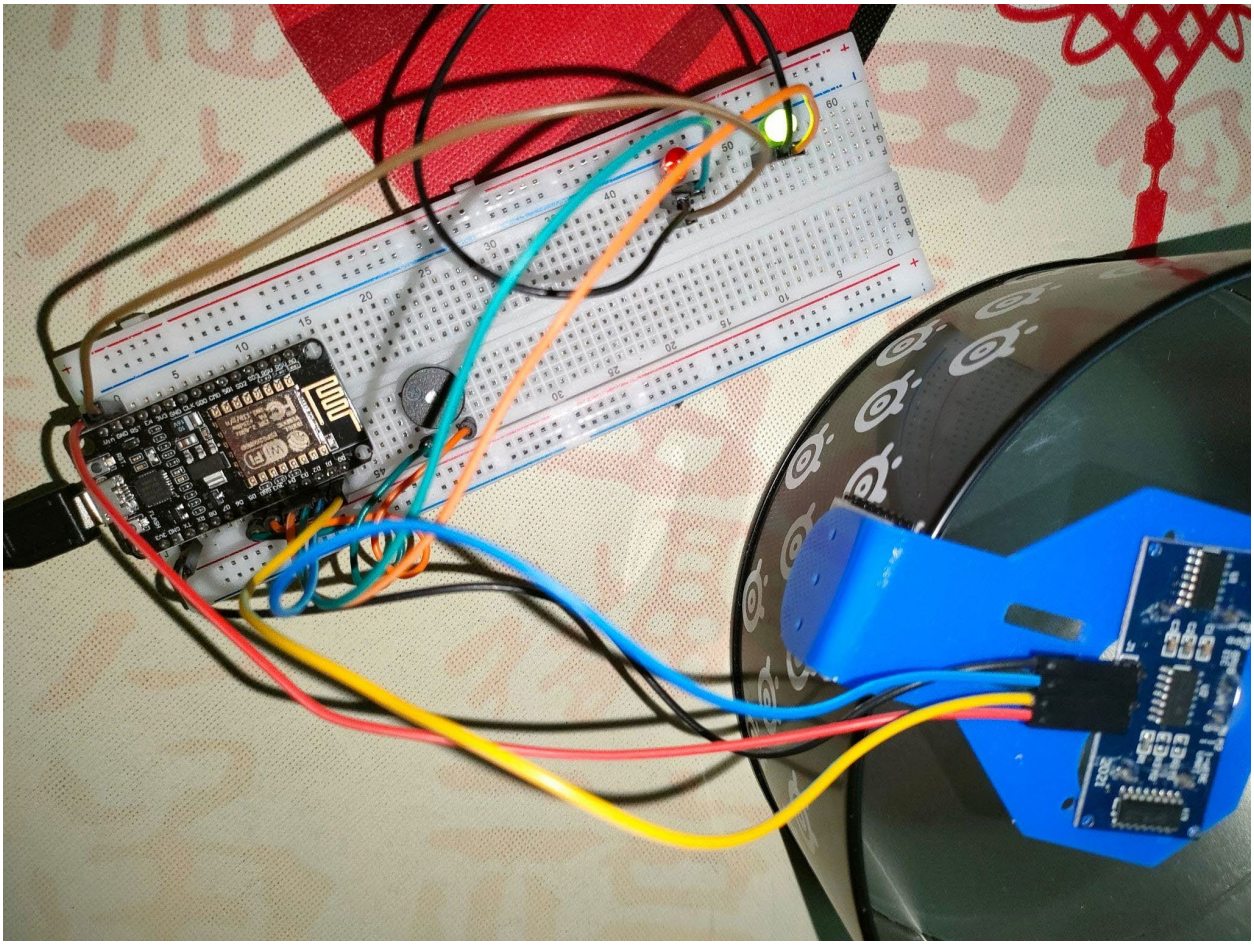

Figure 4 Hardware of the proposed flood monitoring system

Figure 5 shows the interface of the Arduino IoT Cloud on a smartphone. A water level graph is displayed here, with red and green LEDs indicating flood danger and safety, respectively.

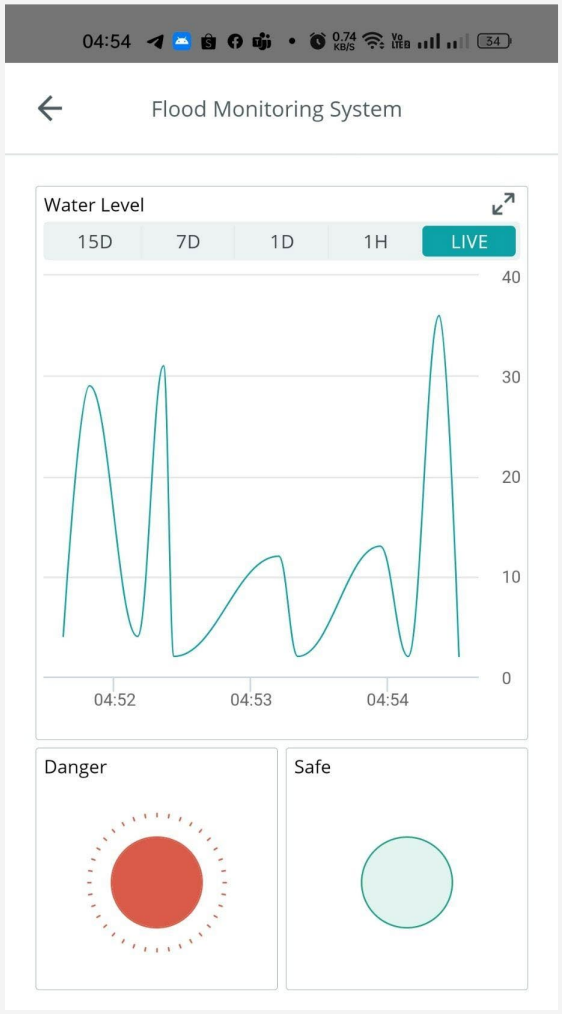

Figure 5 Interface of Arduino IoT Cloud on smartphone