Tampere University of Applied Sciences

Project Report for Weather Station

<Rain 3>

Group name: Group B

Hongxia Wang
Jingjing Yang
Xiaosi Huang

Embedded System
April 2022

Bachelor's Degree in Software Engineering

CONTENTS

ABBREVIATIONS AND TERMS

ADC	Analog/Digital Conversion
IP Address	Internet Protocol Address
ISR	Interrupt Service Routine
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
UI	User Interfaces

1 Overview

The project is to implement an intelligent IoT system, weather station system, combining hardware and software together system.

Two streams are running from January to April:

1) Theory and exercise section: students learn how to implement small exercises that can be used to integrate into the whole weather system.
   The learning contents include introduction to electronics, A/D- conversion, sensor technologies, broker functionality, IoT- protocols, basics of C and C++ microcontroller programming with IoT interfaces, etc.
2) The IoT-weather station project section: teams plan and run their own project from sensors to UI. The project section also needed a web programming course to make UI for visualizing measured sensor values to create data transfer from sensors to cloud and vice versa. Therefore, we students get aware of IoT architecture from beginning to end through this kind of participation.

2 Goals and Scope

2.1 Project goals

The project is to set up a real IoT- weather system on Embedded System and Web Development courses. The signals handled in Arduino based system include wind speed, wind direction, temperature, humidity out, humidity In, rain level, light level and air pressure. Our team is responsible for rain level 3. The project is selling their output (a Weather-app application on a virtual server) to their customers (teachers) and review dates are defined by the customers.

2.2 Project scope

Our project teamwork is responsible to design the application circuit board and implement it with required functionalities. In addition, we need to produce all relevant documentation during the project lifetime. We have been using Arduino, raspberry PI, and backend to frontend to finish the whole project.

3 Project Organization

TAMK builds a weather system on embedded and web development courses. There are different kinds of signals which are handled in Arduino based system.

The teachers manage the course process and guide the students for lab activities. We randomly form a three-person group for teamwork. Each group has one project manager who guides the whole teamwork and project. All group members have own roles and responsibilities as well during the entire process. All team members cooperate together.

3.1 Project team organization

We as a team worked together for most of the activities related to our project, unless someone could not come to campus for lab tasks.

Roles	Personnel	Responsibilities
Project manager	Jingjing Yang	Guide the whole team to deliver our project on time. Regular team meeting on Teams and WeChat.
Technical manager	Xiaosi Huang	Check codes running alright, backup.
Documentation manager	Hongxia Wang	Check all docs required are included, backup.

3.2 Project-internal function

Project-internal Function	Organization: Name
Quality Assurance	TAMK
System Test Lead	TAMK
Validation Lead	Jingjing yang
Configuration Mgmt	Jingjing yang
Change Mgmt	Jingjing,Xiaosi,Hongxia

4 Resources

Human resources	three students as a team, three teachers as coaches as well as customers
Competency resources	Skills of coding, web design, electronic wring, etc.
Equipment resources	TAMK laboratory
Software resources	Arduino, Node, JS...
Time resource	Spring semester 2022, January to April

Different technologies and tools are needed at distinct stages of the project:

- Raspberry forwards the data to REST API using HTTP POST method. API is running on title cloud.
- The receiving API is implemented using Node.js, so we run JavaScript on the server side.
- When the node receives the message, it makes a database connection and runs SQL INSERT command to the database. We use PostgreSQL as a database management system.
- Node and database together form a backend. Fronted is a web application that can display data in a browser.
- The application reads data through API using the HTTP GET method and creates a table or visual representation of the data for the user.

5 Timeline and Schedule

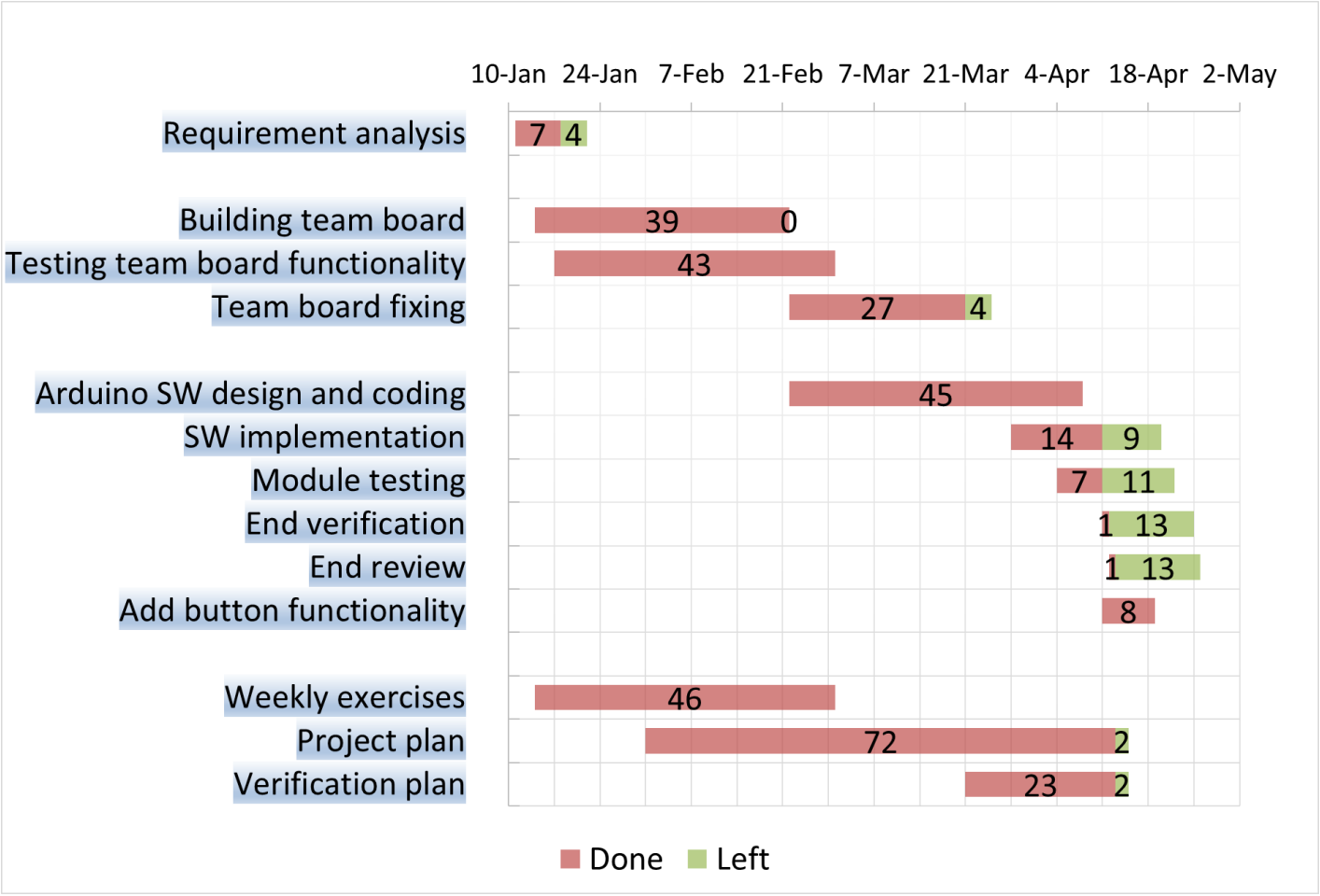
5.1 Project timeline

The creation of weather station project involves several basic activities, and these activities are not strictly linear – they overlap and interact. For weekly plan, we have made a worksheet of project timeline in Excel and a Gantt chart is generated based on that timeline.

Testing team board functionality, Arduino SW design and coding, Add button functionality, and weekly exercises took more days than we planned.

Task	Start Date	End Date	Planned end	Done	Planned	Left
Requirement analysis	2022/1/11	2022/1/18	2022/1/22	7	11	4
Building team board	2022/1/14	2022/2/22	2022/2/22	39	39	0
Testing team board functionality	2022/1/17	2022/3/1	2022/2/26	43	40	-3
Team board fixing	2022/2/22	2022/3/21	2022/3/25	27	31	4
Arduino SW design and coding	2022/2/22	2022/4/8	2022/3/31	45	37	-8
SW implementation	2022/3/28	2022/4/11	2022/4/20	14	23	9
Module testing	2022/4/4	2022/4/11	2022/4/22	7	18	11
End verification	2022/4/11	2022/4/12	2022/4/25	1	14	13
End review	2022/4/12	2022/4/13	2022/4/26	1	14	13
Add button functionality	2022/4/11	2022/4/19	2022/4/15	8	4	-4
Weekly exercises	2022/1/14	2022/3/1	2022/2/22	46	39	-7
Project plan	2022/1/31	2022/4/13	2022/4/15	72	74	2
Verification plan	2022/3/21	2022/4/13	2022/4/15	23	25	2

5.2 Gant-chart



5.1 Milestones

Milestones	Description	Milestone Criteria	Planned Date
M0	Start Project	project goals and scope defined	<2022-02-07>
M1	Start Execution	requirements agreed, resources committed	< 2022-02-15 >
M2	Confirm Execution	Coding of functionality	< 2022-02-25>
M3	Release project	Product system tested and fixed	<2022-04-11>
M4	Close Project	documentation reviewed	<2022-04-25>

5.2 Working hours

People	Activities	Timing	Estimated Working hours
Jingjing Xiaosi, Hongxia	attending theory lessons; absorbing knowledge.	11.1.2022- 7.04.2022	2-3 hour/week for 12 weeks
Jingjing Xiaosi, Hongxia	doing weekly tasks; board wiring; code testing.	14.1.2022- 11.04.2022	2-4 hour/week for 10 weeks
Jingjing	code designing; finalizing documentation.	17.1.2022- 19.1.2022	3-6 hour/week for 6 weeks
Jingjing Xiaosi, Hongxia	learning related skills; teamwork; documentation, etc.	17.1.2022 - 12.1.2022	2-5 hour/week for 8 weeks

6 Development Process

We completed the whole project according to the plan formulated by the teacher on Moodle, combining what we learned from theory lessons and laboratory tasks.

Development Environment

Item	Applied for	Availability by
Methods		
theory + lab	Requirements capturing	M1
Tools		
Laboratory equipment	Design and implementation	M3
Languages		
C++	Arduino design	M2
Node.js, SQL	Web interface	M2

Stages of implementation

1) Arduino SW design.
2) Arduino SW coding.
3) Integrating code with HW.
4) Final testing SW.
5) REST-API / Database design and implementation (Gathering weather data and pushing the data to database).
6) Web UI design and coding.
7) Commercial use and market distribution (Excluded).
8) Experiment cost and human resource cost (Excluded).

7 Risk Management

To ensure the completion of the project on the Weather Station, our team members need to start the project according to the project date. Project content planning should be done in advance. Team discussion of the project schedule is also particularly important. The roles of each team member should be clearly defined.

7.1 Risk assessment standard

Threat Likelihood	**Impact**		
	Low (10)	**Medium (50)**	**High (100)**
Low (0.1)	Low Risk (1)	Low Risk (5)	Low Risk (10)
Medium (0.5)	Low Risk (5)	Medium Risk (25)	Medium Risk (50)
High (1.0)	Low Risk (10)	Medium Risk (50)	High Risk (100)

Risk Scale: Low (1 – 10); Medium (10 - 50); High (50 - 100).

7.2 Risk management

No.	**Risk**	**Likelihood**	**Impact**	**Risk Rating**	**Risk Management**
01	Project operation is not running smoothly	Medium	Medium	Medium Risk	#Task requirements clarification #Equal distribution on weekly tasks #All member highly involved in the task
02	Project goal is not clear	Medium	High	Low Risk	#The overall plan of the project #Weekly tasks accomplishment goal #Fully focus on the current project
03	Project time management is not on time	Medium	Medium	Medium Risk	#Time spending on task allocation #Time sending on task accomplishment #Efficient performance
04	Waste on Project cost management	Low	Medium	Low Risk	#Laboratory recourse without waste #Online free cost recourses #Recycle consciousness on current materials
05	Project quality control	High	Medium	Medium Risk	#Quality control on every single tasks #Team members take turn to check #Some help from tutors

06	Teamwork non-cooperation	Low	High	Low Risk	#Mutual understanding #Mutual help #Mutual support
07	Ineffective team communication	Low	High	Low Risk	#Communication promptly #Information sharing timely #Recording correctly
08	Team board missing	Medium	High	Medium Risk	# Marking the board #Keeping the board in the Lab's fixed position #Taking its location picture as records #Knowing the principle for wiring for rebuilt
09	File missing	Low	High	Low Risk	#All member keeps the synchronization file #Backup via OneDrive
10	Codes fail to function	High	High	High Risk	#Self-learning #Mutual help
11	Team members sick or cannot come to the Lab	Medium	Medium	Medium Risk	#Teamwork #Distance working
12	Laboratory unavailable due to corona	Medium	Medium	Medium Risk	#Remote learning #Self-learning # Web-based learning resources
13	Lab safety	Medium	Medium	Medium Risk	#Be strict with the lab rules #Power dump

8 Communication and Reporting

One of the key factors that makes a project challenging is ineffective communication. And it is a risk that should not be taken lightly. Any mistakes caused by ineffective communication would lead to our members being stuck on the wrong testing operation and time wasting. Once there is any updated information in the team, team members should also timely communicate and exchange information. The team members who record the data also do an excellent job of recording and managing the data so that the correct data is shared within the team.

Communication type	**Method / Tool**	**Frequency/ Schedule**	**Information**	**Participants / Responsible**
Internal Communication:				
Project Meetings	By Teleconfer-ence WeChat or Email	Weekly and on event	Project status, problems, risks, changed requirements	Project Mgrs Project Team
Sharing of project data	Shared Project Server	When available	All project documentation and reports	Project Mgrs Project Team Members
Milestone Meetings	Teleconference	Before milestones	Project status (progress)	Project Mgrs Sub-project Mgrs
External Communication and Reporting:				
Project Report	Office software	Monthly	Project status - progress - forecast - risks	Project Manager Sub-Project Man-agers

9 Delivery Plan

- Deliverables: system code and documentation files
- Receivers: teachers
- Date: 19th April 2022

10 Revision

Version	Chapter (C)	Date
1st	C3, C5, C7	Presented 22nd, Feb
2nd	C2, C6, C8	Presented 11th, April
3rd	C1, C4, C9	Submitted 19th, April

Tampere University of Applied Sciences

Technical report of IoT weather system

Group B
Hongxia Wang
Jingjing Yang
Xiaosi Huang

Embedded System
April 2022

Bachelor's Degree in Software Engineering

CONTENTS

1 Introduction

This document demonstrates the required application circuit functionality that are related to our application circuit and the verification against the requirements.

Application circuit is converting the signal from the signal box to be suitable for Arduino input. Each team is responsible for one signal, analog or digital. And each team is responsible to design the application circuit and implement it. MQTT broker/ C++ client is receiving measured value from all teams.

The signal our team responsible for is Rain3. It is a digital signal type, and D3 is used as the digital input pin.

2 Application circuit board

2.1 Components

Arduino Mega 2560

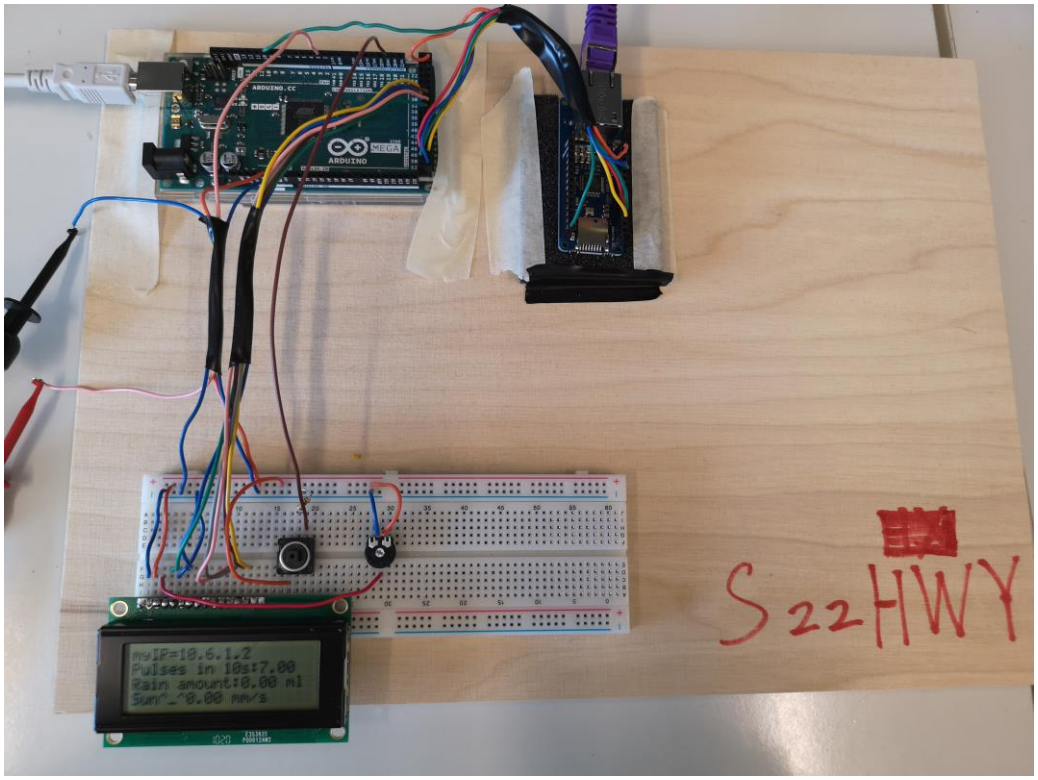
Breadboard

20x4-sized LCD display

A potentiometer (to adjust LCD brightness)

Ethernet W5500 Blue module

A button and a 10kΩ resistor (using D19 as input pin)



2.2 Circuit

Connection between LCD interface pin and Arduino pin number:

LCD pin	RS	Enable	D4	D5	D6	D7	R/W
Arduino pin	37	36	35	34	33	32	ground

For more detailed wiring methods, please refer to the appendixes attached.

2.3 Functionality

Required:

1) Fetch IP number and print it on the LCD display first row.
2) Read digital signal from rain sensor.
3) Count the pulses in every 10 seconds and print that value on the LCD display second row.
4) Convert the received value to an average rain amount collected *in an area of 55 cm^2 within a period of 10 seconds*, in the unit of milliliter per second (ml/s)
5) Print the value in (3) on the LCD display third row.
6) Send sensor value in (3) to MQTT broker every 10 seconds.

Extra:

7) Calculate the rainfall in the unit o*f cubic millimeter per square millimeter* (mm)
8) Indicate the rain level (sunny, light rain, medium rain, heavy rain) based on rain depth (0,0-10,10-50,50-) in (7).
9) Print the value in (6) and (7) on the LCD display fourth row.
10) Turn on the Arduino built-in led if the rain is heavy level.
11) Print group name and measured signal on the LCD display fourth row when the button is pressed, and let the information stay there for five seconds, then prints back our IP number, until the button is pressed again.

However, that kind of definition of rainfall in (8) may have little meaning in real world. For example, if we set 0.1 Hz output from the frequency generator, it means one pulse in 10 seconds, which is the smallest value our system can take in the period we set. Thus, 3ml water for a collecting area of 55cm^2 (heavy enough rain) results in 3000mm^3/(10s*5500mm^2) = 0.055 mm/s = 196 mm/h. This actually is violent rain in real (please see the contents below which are taken from internet), but in our system, it is defined as light rain.

The reason why we define 0-10mm/s as light rain and 10-50mm/s as medium rain is that we want to test if our code runs all right from learning purpose. For our sensor, one sensor tip over gives one pulse, so the sensor accuracy will be

improved if for one tip over water is needed only 3 microliters (1μL=0.001mL) instead of 3 mL. Or we set the pulse counting period longer, for example 1 hour instead of 10 seconds.

“Rainfall refers to quantity of rain falling within a given area in a given time and is often expressed in millimeters per day (mm / day) which represents the total depth of rainwater (mm) during 24 hours. “

“The following categories are used to classify rainfall intensity:
Light rain — when the precipitation rate is < 2.5 mm per hour
Moderate rain — when the precipitation rate is between 2.5 mm – 7.6 mm or 10 mm per hour
Heavy rain — when the precipitation rate is > 7.6 mm per hour, or between 10 mm and 50 mm per hour
Violent rain — when the precipitation rate is > 50 mm per hour”

3 Functionality verification

3.1 Signal waveform

Square wave;

Voltage level 0V and 5V;

Low level “0” = 0.0V

High level “1” = 5.0V

3.2 Calculation

- One sensor tip over gives one pulse.
- For on tip over water is needed 3 ml (3000 mm^3).
- Collecting area is 55 cm^2 (5500 mm^2).
- Counting period 10s.

Rain amount of 55 cm^2 in 10 seconds = pulse*3 <ml>

Average one: Rain amount of 55 cm^2 per second= (pulse / 10.0) * 3 <ml>

// (mm^3/mm^2=mm)

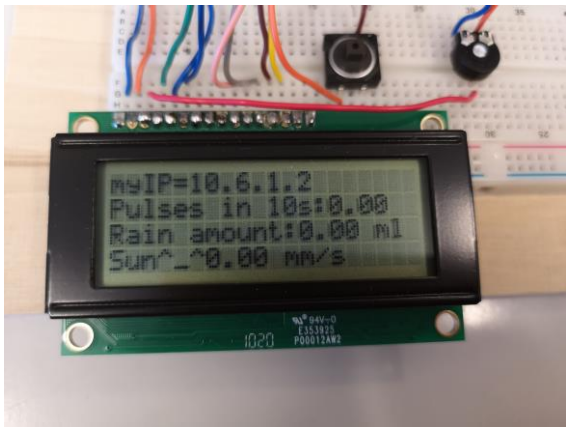
Average two: rain depth per cm^2 per second = ((pulse / 10.0) * 3000) / 5500 <mm>

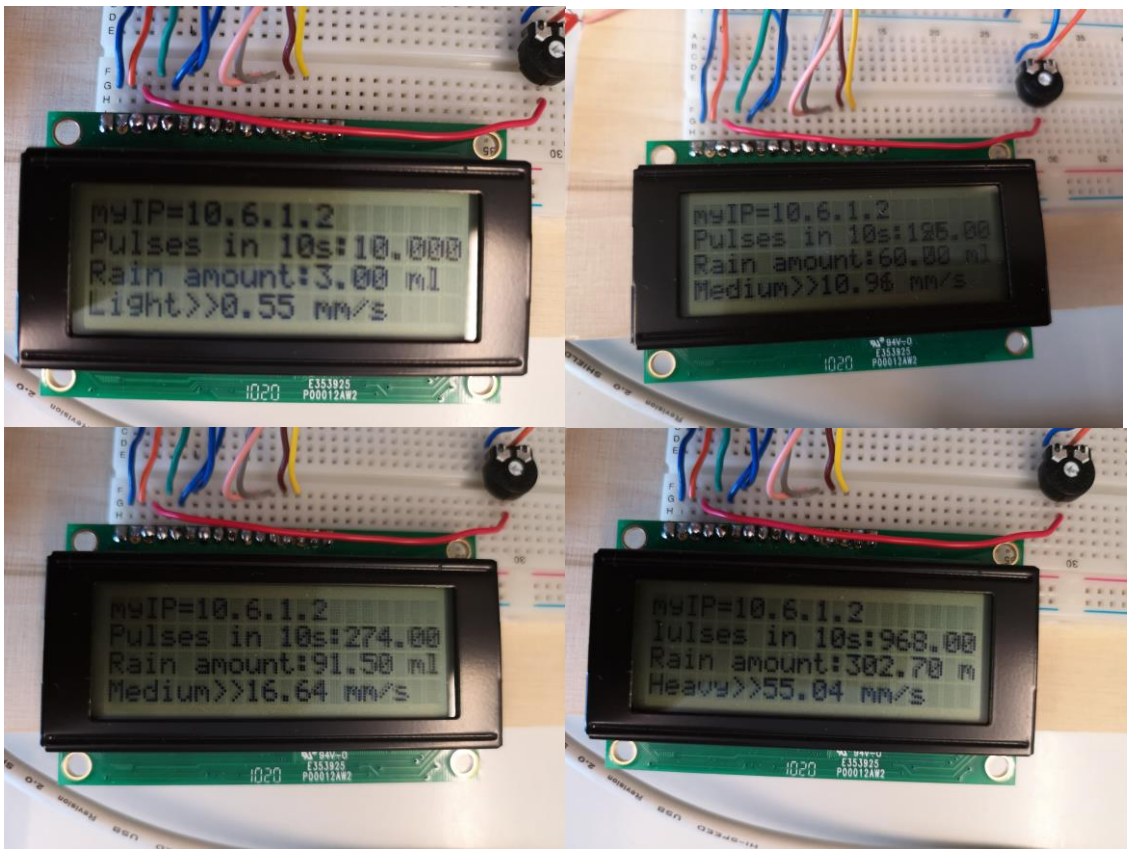
Frequency output (Hz)	Pulses in 10s	Average rain amount ml/(55cm^2*s)	Rainfall mm/s	Rain level	Built-in LCD
0	0	0	0	Sun	off
1.0	10	3.00	0.55	Light	off
20	200	60.00	10.96	Medium	off
30.5	305	91.50	16.64	Medium	off
100.7	1007	302.1	55.04	Heavy	on

3.1 LCD-display

Local LCD-display content consists of IP number, counted pulses, measured values in two different units, and rain level. The code has been tested with a signal generator.

- Verification for functionalities 1-5 & 7-9

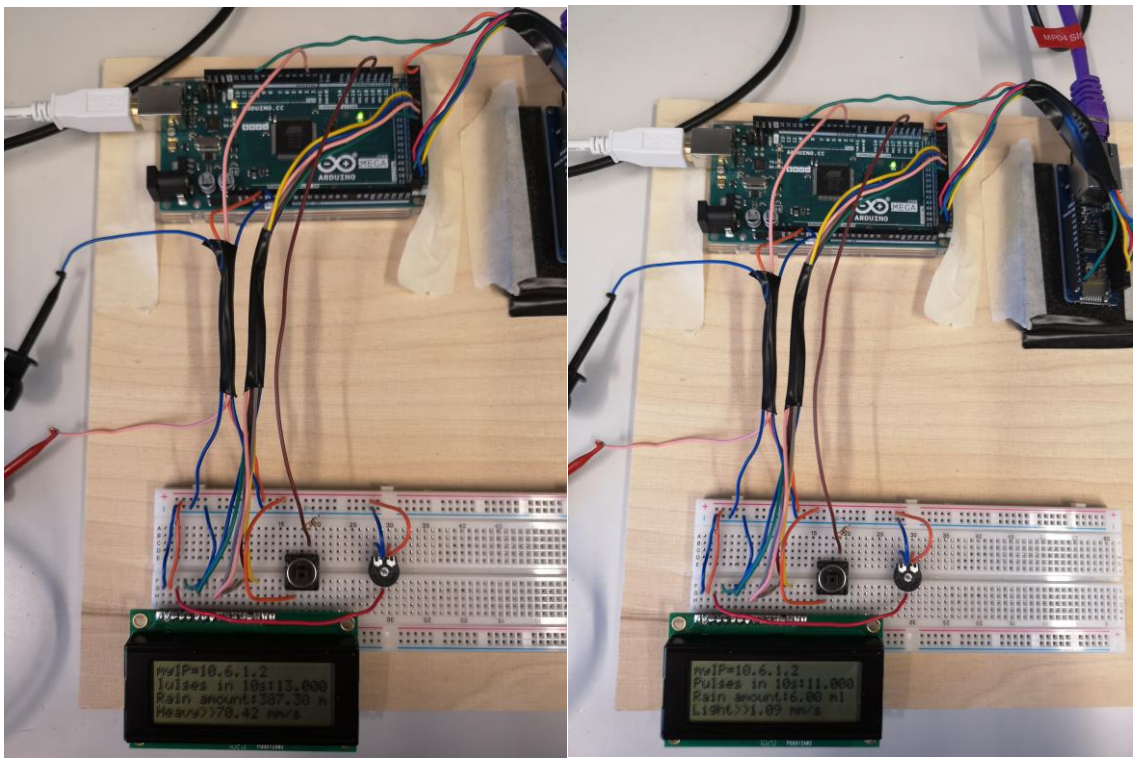




3.2 Built-in LED

- Verification for functionality 10

Built-in LED is turned on when rain level is heavy (left picuture).



3.1 Button-pressed

- Verification for Functionality 11

3.2 Message to MQTT broker

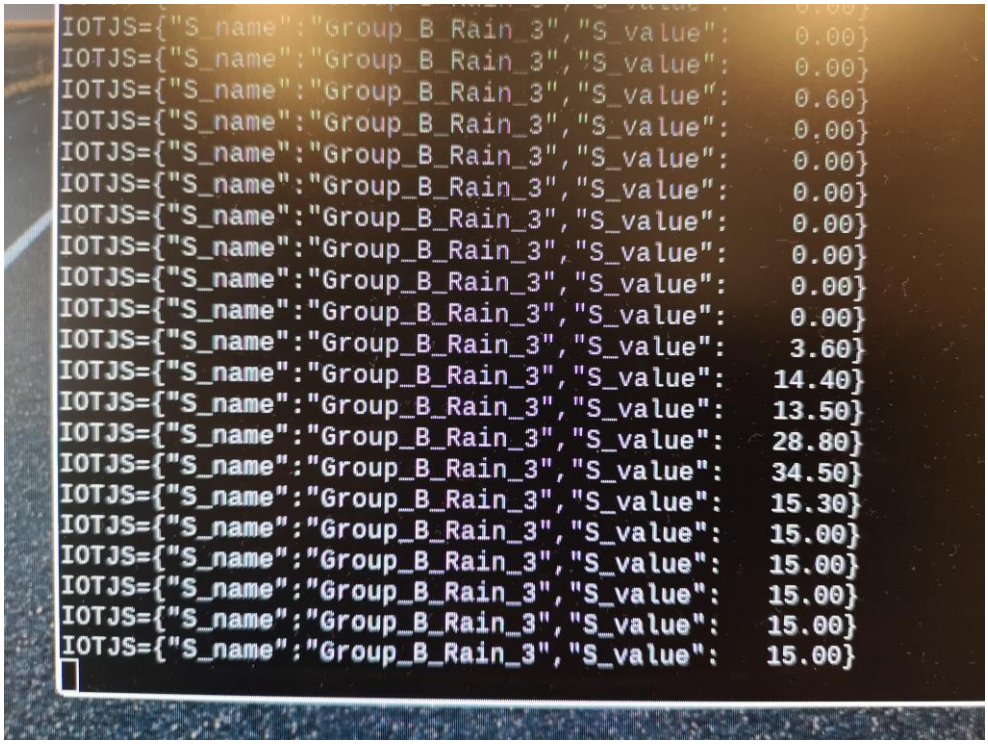
- Verification for Functionality 6

Message is sent in each 10 seconds.

The value sent is accurate to two decimal places.

Example:

IOTJS={"S_name":"Group_B_Rain_3","S_value": 15.00}



3.3 Message to Raspberry

Example:

========== Message received ==========

Topic = ICT4_out_2020

Message = {"S_name":"Group_B_Rain_3","S_value": 32.70}

JSON Message = {"S_name":" Group_B_Rain_3","S_value": 32.70}

Measurement : Rain_3 = 32.700001

Parsing done OK

JSON to db = {" Group_B_Rain_3":32.700001}

POST with curl = {"device_id":"ICT_2021","data ": {" Group_B_Rain_3":32.700001}}

HTTP/1.1 201 Created

Server: nginx/1.21.5

date: Tue, 19 Apr 2022 08:35:46 GMT

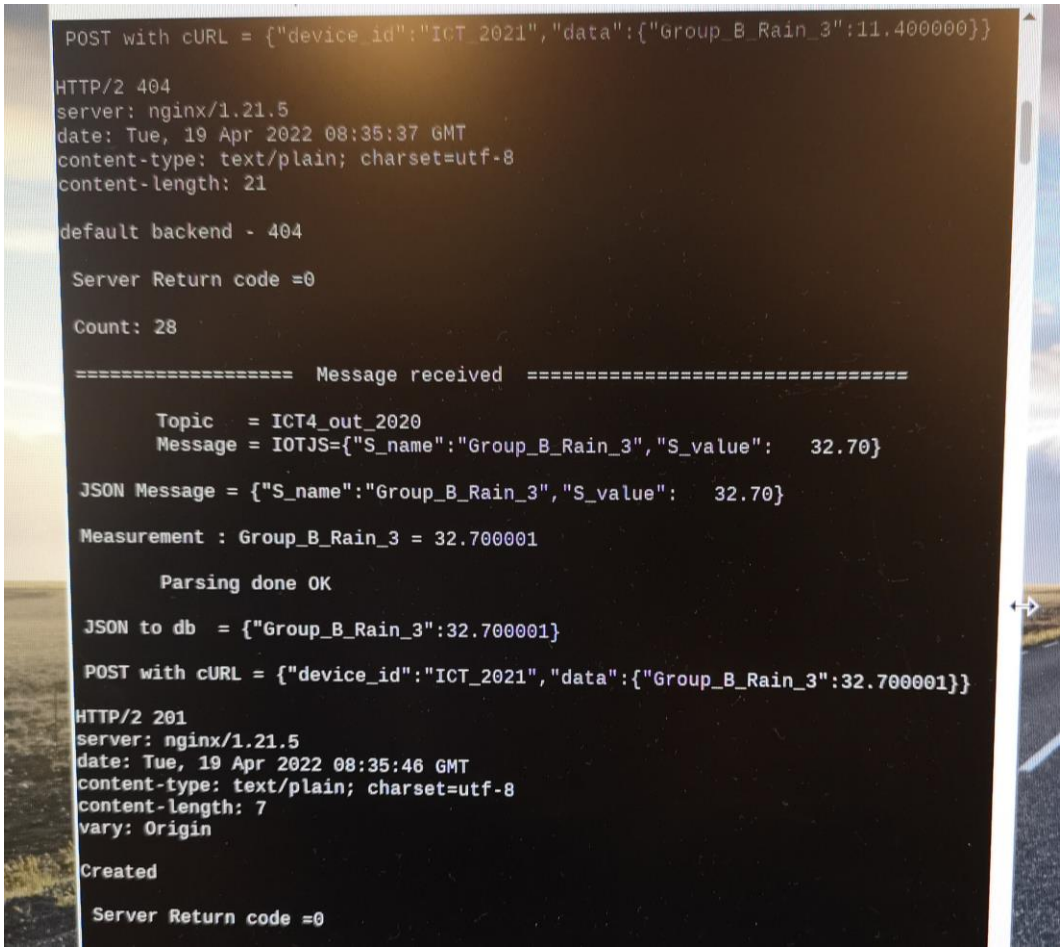
content-type: text/plain; charset=uft-8

content-length:7

vary: Origin

Created

Server Return code =0



4 Arduino source code

```
//////////////////////////////////// Libraries included/////////////////////////////////////
#include <LiquidCrystal.h>                          // include LCD library

#include <Ethernet.h>                               // include Ethernet library

#include <PubSubClient.h>                           // include MQTT library

#include <TimerOne.h>                               // include timer library

////////////////////////// LCD pin wiring settings for MEGA ///////////////////////////////
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
//               RS  E   D4  D5  D6  D7
LiquidCrystal lcd(37, 36, 35, 34, 33, 32);

EthernetClient ethClient;                           // Ethernet object var


////////////////////////////////// MAC number ////////////////////////////////////////////
static uint8_t mymac[6] = { 0x44, 0x76, 0x58, 0x10, 0x00, 0x73 };


////////////////////////////////// MQTT settings /////////////////////////////////////////
unsigned int Port = 1883;           // MQTT port number
byte server[] = { 10, 6, 0, 21 };   // TAMK IP

char* deviceId  = "2020a72145";     // set device id (MQTT client username)
char* clientId     = "a771345";     // set a random string (max 23 chars, MQTT client id)
char* deviceSecret = "tamk1";       // set device secret (MQTT client password)

//////////////////////////////// MQTT Server settings ////////////////////////////////////
// subscription callback for received MQTTT messages
void callback(char* topic, byte* payload, unsigned int length);
// mqtt client
PubSubClient client(server, Port, callback, ethClient);


//////////////////////////////// MQTT topic names  //////////////////////////////////////
#define inTopic    "ICT4_in_2020"             // * MQTT channel where data are received
#define outTopic   "ICT4_out_2020"            // * MQTT channel where data is send
////////////////////////////////////////////////////////////////////////////////////////

///****************************** Variables *****************************************
// constants won't change. They're used here to set pin numbers:
const int sensorPin = 3;                // number of the digital signal pin
const int buttonPin = 19;               // number of the pushbutton pin

// variables will change:
int buttonState = 0;                    // variable for reading the pushbutton state
// declaring variables volatile as they are used in ISR section
volatile double pulse = 0;              // counts the pulses
volatile double i_time = 0;
// set datatype double to keep precision
double  rainLevel = 0;                  // average rain amount,unit ml/area, area=55cm2
double rainfall = 0;                    // average rain depth, unit mm/s
///**********************************************************************************

```

```
58  ////////////////////////////// SETUP section /////////////////////////////////////////////
59  ////////////////////////////// SETUP section /////////////////////////////////////////////
60  ////////////////////////////// SETUP section /////////////////////////////////////////////
61  void setup() {
62    // put your setup code here, to run once:
63
64    Serial.begin(9600);                          // Serial monitor baudrate  = 9600
65
66    lcd.begin(20, 4);                            // Display size definition 20 char 4 rows
67
68    lcd.setCursor(0, 0);                        // set cursor to left upper corner
69    //         01234567890123456789
70    lcd.print("30.3.2020 Alyk jatk ");          // print to lCD
71
72    Serial.println("Start 30.3.2020");          // print to serial monitor
73
74    delay(500);
75
76    fetch_IP();                                 // initialize Ethernet connection
77
78    Connect_MQTT_server();                      // connect to MQTT server
79
80    ///*****************************************************************************************
81    pinMode(sensorPin, INPUT);       // initialize sensorPin as an input for signal sensing
82    pinMode(buttonPin, INPUT);       // initialize buttonPin as an input for button state
83
84    // With a function called attachInterrupt(), we can execute an ISR
85    // when the state of the Arduino io-pin changes.
86    // Attach an interrupt to the ISR vector // Pin 3, Routine:sensor_ISR, RISING Edge
87    attachInterrupt(digitalPinToInterrupt(sensorPin), sensor_ISR, RISING);
88
89    // initialize timer1, and set one-half second as a period
90    Timer1.initialize(500000);
91    //attaches callback() as a timer overflow
92    Timer1.attachInterrupt(callback);
93    ///*****************************************************************************************
94  }
95
96
97  ///************************ LOOP section ***************************************************
98  ///************************ LOOP section ***************************************************
99  ///************************ LOOP section ***************************************************
100 void loop()
101 {
102   // read the state of the pushbutton value:
103   buttonState = digitalRead(buttonPin);
104   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
105   if (buttonState == HIGH) {
106     lcd.setCursor(0, 0);                  // set cursor to first row
107     lcd.print("Group B-->Rain 3    ");     // print group name and signal if button pressed
108     // delay a little bit to improve simulation performance
109     delay(5000);
110     lcd.setCursor(0, 0);                  // set cursor to first row
111     //         01234567890123456789
112     lcd.print("                    ");   //clear the first row to print myIP
113
114     lcd.setCursor(0, 0);                // set cursor to 1st row
115     lcd.print("myIP=");
116     lcd.print(Ethernet.localIP());      // print IP number from DHCP server again
117     delay(1500);
118   }
119
```

```
  lcd.setCursor(0, 1);                          // set cursor to 2nd row
  //         01234567890123456789
  lcd.print("Pulses in 10s:");
  lcd.print(pulse);                            // print to LCD the pulses per 10 seconds

  lcd.setCursor(0, 2);                 // set cursor to 3rd row
  //         01234567890123456789
  lcd.print("                    ");  //clear the 3rd row to print updated rain amount
  lcd.setCursor(0, 2);                 // set cursor to 3rd row
  lcd.print("Rain amount:");
  lcd.print(rainLevel);               // print value of rain amount to LCD
  lcd.print(" ml");

  lcd.setCursor(0, 3);                   // set cursor to 4th row
  //         01234567890123456789
  lcd.print("                    ");    //clear the 4th row to print updated rain amount
  lcd.setCursor(0, 3);                   // set cursor to 4th row
  if (rainfall == 0) {                   // print the rain Level
    lcd.print("Sun^_^");
    digitalWrite(LED_BUILTIN, LOW);// turn the built-in LED off by making the voltage LOW
  }
  else if (rainfall <= 10) {
    lcd.print("Light>>");
    digitalWrite(LED_BUILTIN, LOW);
  }
  else if (rainfall <= 50) {
    lcd.print("Medium>>");
    digitalWrite(LED_BUILTIN, LOW);
  }
  else {
    lcd.print("Heavy>>");
    digitalWrite(LED_BUILTIN, HIGH);// turn the built-in LED on (HIGH is the voltage level)
  }
  lcd.print(rainfall);                // print value of rain depth to LCD
  lcd.print(" mm/s");
  delay(1000);
}

///************************ Interrupt service routine ************************************
// This is an ISR that counts the pulses.
// code to be executed for each occurred pulse
// Rising endge of pin voltage will activate interrupt service routine
void sensor_ISR() {
  pulse++;                        //increment variable "pulse"
}

///************************ Timer interrupt settings ************************************
///************************ Timer interrupt settings ************************************
///************************ Timer interrupt settings ************************************
// Interrupt routine called in each 0.5 sec by HW timer
void callback() {
  i_time++;                                  //increment variable "i_time" in each 0.5 sec

  if (i_time >= 20) {
    //reset value of variable "i_time" to ZERO for the next 10 sec
    i_time = 0;

    rainLevel = (pulse / 10.0) * 3;             //save rain amount to variable "rainLevel"
    // send  MQTT message every 10 seconds
    send_MQTT_message("Group_B_Rain_3", rainLevel);

```

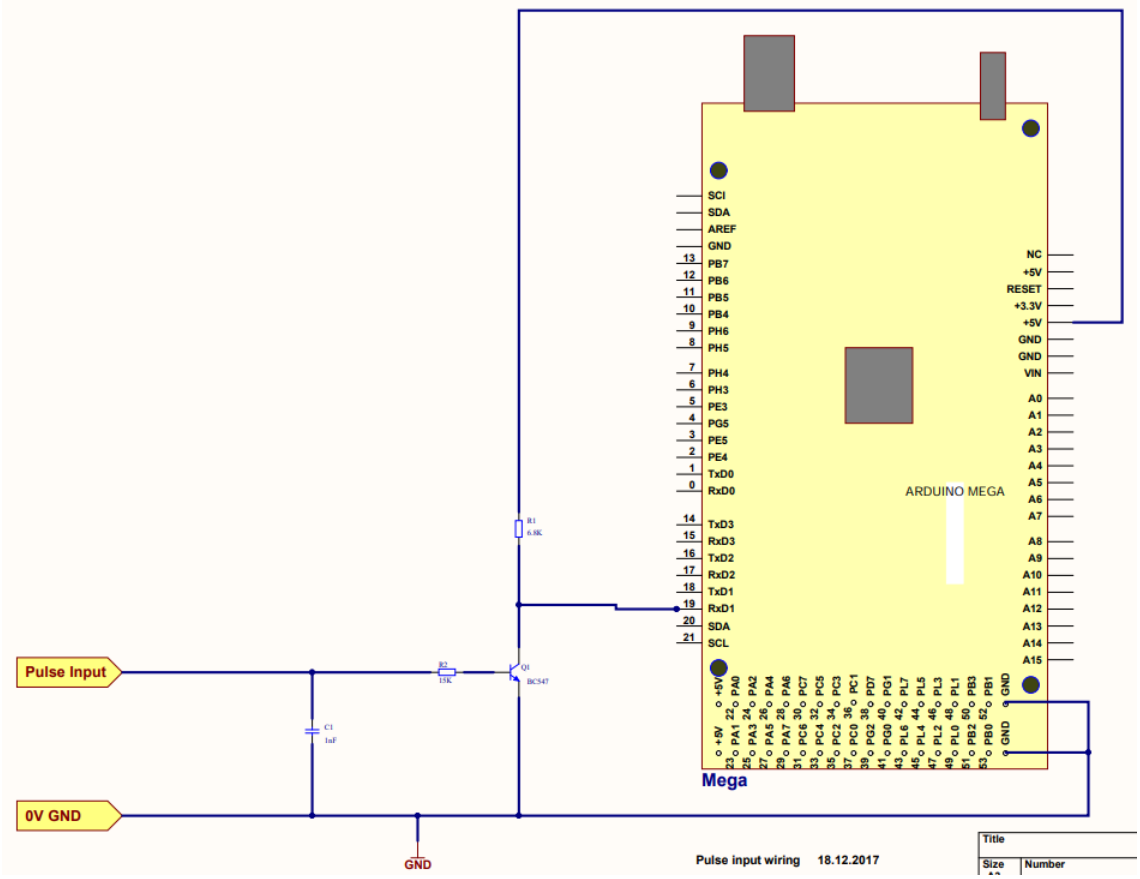
```
    //collection area 55cm2(5500mm2),period 10s
    rainfall = ((pulse / 10.0) * 3000) / 5500; //save rain depth to variable "rainfall"

    // reset value of variable "pulse" to ZERO for the next 10 sec
    pulse = 0;
  }
}


////////////////////////////// Ethernet routine section ////////////////////////////////
void fetch_IP(void)
{
  byte rev = 1;



  lcd.setCursor(0, 0);

  //         01234567890123456789
  lcd.print("     Waiting IP     ");

  rev = Ethernet.begin( mymac);            // get IP number from DHCP server

  Serial.print( F("\nW5100 Revision ") );

  if ( rev == 0) {

    Serial.println( F( "Failed to access Ethernet controller" ) );

    // 0123456789012345689
    lcd.setCursor(0, 0); lcd.print(" Ethernet failed   ");
  }


  Serial.println( F( "Setting up DHCP" ));
  Serial.print("Connected with IP: ");
  Serial.println(Ethernet.localIP());


  lcd.setCursor(0, 0);
  //         01234567890123456789
  lcd.print("                    ");

  lcd.setCursor(0, 0);                  // set cursor to 1st row
  lcd.print("myIP=");
  lcd.print(Ethernet.localIP());        // print IP number from DHCP server
  delay(1500);

}
```

```
232 ////////////////////////////// MQTT routine section /////////////////////////////////////
233
234 void send_MQTT_message(String S_name, double rainLevel)
235 {
236   char bufa[50];
237
238   char str_s_name[20];//  hold The Convert Data
239   S_name.toCharArray(str_s_name, S_name.length() + 1);
240
241   char TempString[20];  //  hold The Convert Data
242   dtostrf(rainLevel, 8, 2, TempString);
243
244
245   // create message with header and data
246   // sprintf(bufa,"IOTJS={\"S_name\":\"group_name_here\",\"S_value\":%s}",str_signal_value);
247   sprintf(bufa, "IOTJS={\"S_name\":\"%s\",\"S_value\":%s}", str_s_name, TempString);
248
249   Serial.println( bufa );            // Print message to serial monitor
250
251   if (client.connected())            // send message to MQTT server
252   {
253     client.publish(outTopic, bufa);
254   }
255   else                               // Reconnect if connection is lost
256   {
257     delay(500);
258
259     lcd.setCursor(0, 1);
260     //         01234567890123456789
261     lcd.print("   RE Connecting    ");
262
263     Serial.println(" RE Connecting" );
264
265     client.connect(clientId, deviceId, deviceSecret);
266
267     delay(1000);                    // wait for reconnecting
268   }
269
270 }
271
272
273
274 ///  MQTT server connection
275
276 void Connect_MQTT_server()
277 {
278   Serial.println(" Connecting to MQTT" );
279
280   // Print MQTT server IP number to Serial monitor
281   Serial.print(server[0]); Serial.print(".");
282   Serial.print(server[1]); Serial.print(".");
283   Serial.print(server[2]); Serial.print(".");
284   Serial.println(server[3]);
285
```

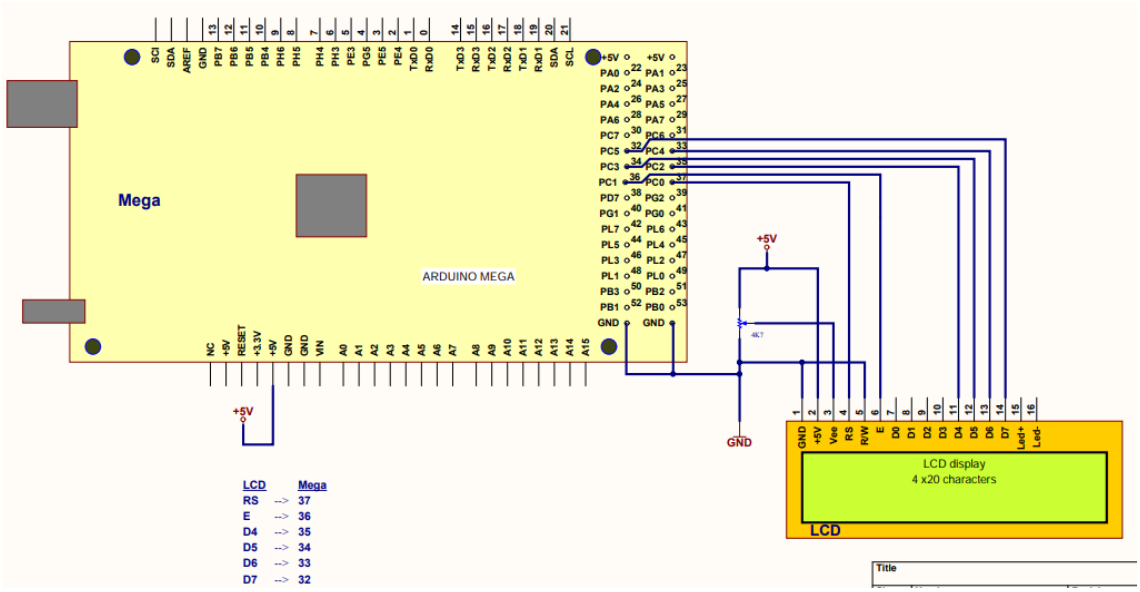
```
  lcd.setCursor(0, 1);
  //         01234567890123456789
  lcd.print("                    ");

  lcd.setCursor(0, 1);
  lcd.print("MQTT=");
  lcd.print(server[0]); lcd.print(".");          // Print MQTT server IP number to LCD
  lcd.print(server[1]); lcd.print(".");
  lcd.print(server[2]); lcd.print(".");
  lcd.print(server[3]);

  delay(500);

  if (!client.connected())                       // check if allready connected
  {
    // connection to MQTT server
    if (client.connect(clientId, deviceId, deviceSecret))
    {
      lcd.setCursor(0, 1);
      lcd.print("Conn");                         // Connection is OK
      Serial.println(" Connected OK " );

      client.subscribe(inTopic);                 // subscript to in topic
    }
    else
    {
      lcd.setCursor(0, 1);
      //         01234567890123456789
      lcd.print("  MQTT Error        ");         // error in connection
      Serial.println(" MQTT Connection ERROR " );
    }
  }
}


/// Receive incoming MQTT message

void callback(char* topic, byte * payload, unsigned int length)
{
  // copu the payload content into a char*
  char* receiv_string;
  receiv_string = (char*) malloc(length + 1);
  memcpy(receiv_string, payload, length);        // copy received message to receiv_string
  receiv_string[length] = '\0';

  lcd.setCursor(0, 0);
  //         01234567890123456789
  lcd.print("Mess=               ");

  lcd.setCursor(5, 0);
  lcd.print(receiv_string);                      // print reveived message to LCD
  Serial.println( receiv_string );

  free(receiv_string);
}
```

APPENDICES

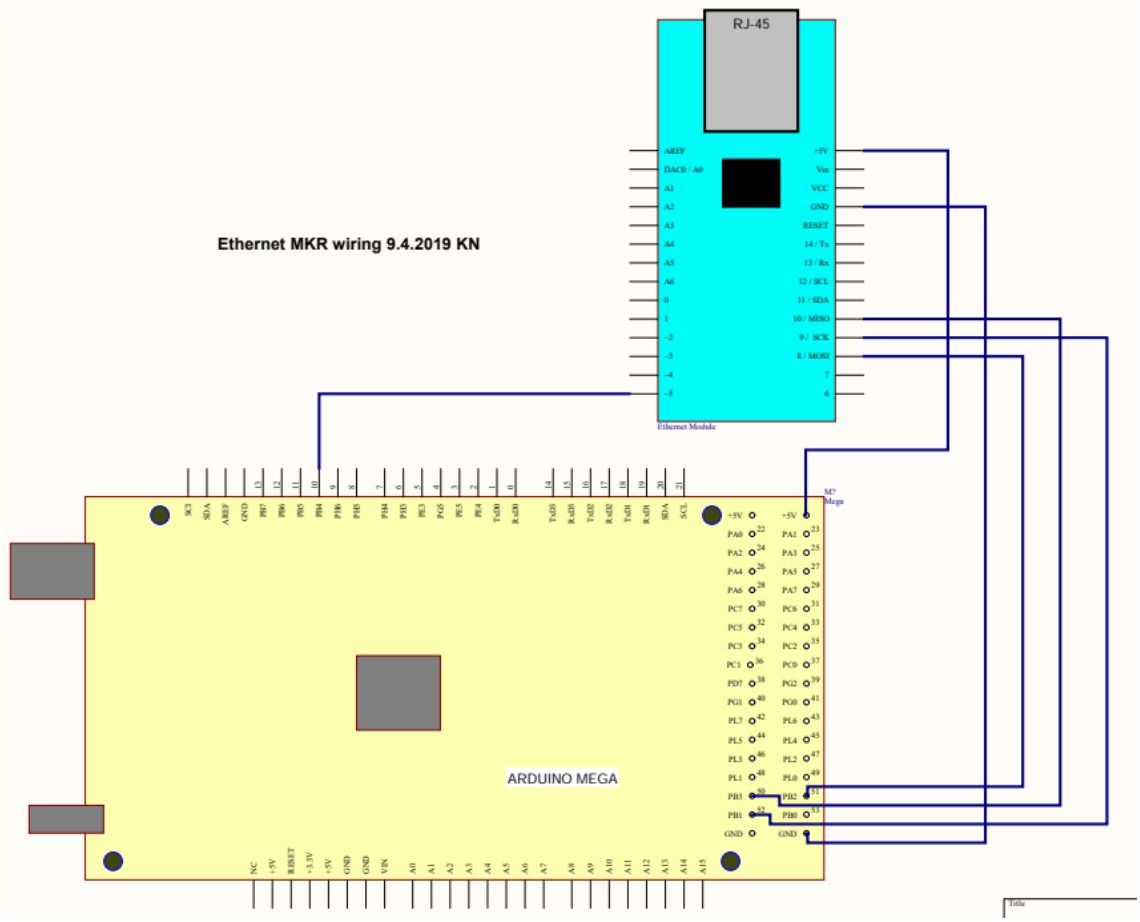
Appendix 1. Arduino wiring



Appendix 2. LCD wiring



Appendix 3. Ethernet Blue wiring



Appendix 4. Button wiring