



TECHNICAL DOCUMENT

ARDUINO WEATHER STATION

HUMIDITY SIGNAL



Authors:

Yen Tran

Israt Sumiya

Hasan Mahmud

APRIL 23, 2020

Finland

Contents

Tables and Figures	1
1. INTRODUCTION	2
2. PROJECT GOALS.....	2
3. USED TECHNOLOGIES IN GENERAL	2
4. SOFTWARE FLOW DIAGRAM	2
4.1. General diagram	2
4.2. Circuitry	3
○ Microcontroller	3
○ Lcd - Liquid Crystal Display	5
○ Ethernet Module Arduino.....	7
○ MQTT CLIENT TO MQTT BROKER	8
○ MQTT SERVER TO REST API	11
5. REFERENCES	13

Tables and Figures

Figures:

Figure 1: WeatherStation Project diagram

Figure 2: Arduino Mega

Figure 3: LCD display

Figure 4: LCD diagram

Figure 5: Code Example LCD

Figure 6: Ethernet module Arduino

Figure 7: Wiring Ethernet module to Arduino mega

Figure 8: Flowchart from MQTT client to server

Figure 9: Software Flow chart of C++ gateway

Tables:

Table 1: Wiring Table LCD

Table 2 Wiring Table Ethernet

1. INTRODUCTION

WeatherStation Arduino is one of the most significant projects of Kakashi Solutions Ltd. This Technical document will introduce and guide users how the software monitors from the board to the web.

2. PROJECT GOALS

Project is created to meet the current necessary of weather data in Tampere, specifically around TAMK 's surrounding area. We aim to bring the most accurate weather information with a friendly user interface therefore users can update their own weather news at home via our website without spending so much money as well as time. Throughout this document, the functionalities and method of working are presented.

3. USED TECHNOLOGIES IN GENERAL

Different technologies and tools are utilized in producing the project, however in general, here is the listed crucial technologies used:

- Arduino
 - Micro controller Arduino Mega
 - Circuitry requires LCD, rheostat
 - Ethernet shield W5100 is used
- Raspberry PI/Virtual Machines
 - Run as IoT gateway, when sending messages Raspberry will forward the messages
 - MQTT broker named Mosquitto and C++ code to construct and forward the http POST messages to API
- Backend
 - API is implemented with Node.js and PostgreSQL is a data repository
 - API and database run in Docker containers on the server
- Frontend
 - Web application works with user's browser which is implemented with HTML, CSS and JavaScript
 - Frontend is on the same server as API and has its own public URL which is also run on a Docker container server
 - There are 4 containers running: API, Database, Adminer (Database management Tool) and Webapp (Frontend Application)

4. SOFTWARE FLOW DIAGRAM

4.1. General diagram

The whole idea of how the system work is described as the picture below. Initially, there are sensors and actuators attached in the rooftop of TAMK. In order to capture the signals received from sensors, there must go through a preprocessing system controlled by a microcontroller board. After doing some transformation and conversion of signals. RaspberryPi/ Virtual Machines acts as an IoT gateway which has MQTT broker plays as an intermediate communicator receiving and sending data forward. Once the data is successfully pushed inside the Database server, web application is created to retrieve the data and visualize them in the easier understanding interface.

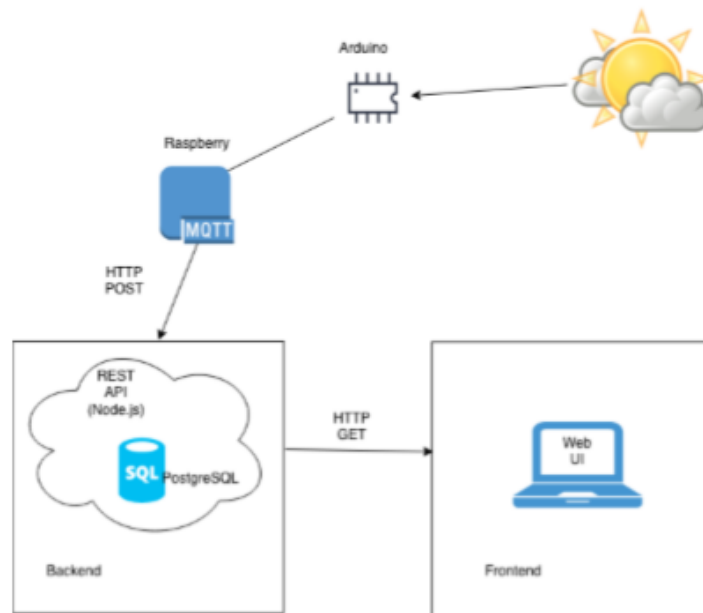


Figure 1 WeatherStation Project - source TAMK

4.2. Circuitry

The board is the main part where can receive and convert the signal before pushing it back to MQTT broker

The board contains: Arduino Mega, LCD with rheostat, Ethernet module

○ Microcontroller

Arduino is a kit for building digital devices. It consists of single board microcontrollers and microcontrollers kit. The boards are equipped with sets of digital and analog input/output pins that may be interfaced to various expansion boards or breadboards and other circuits. The boards feature serial communications interfaces, including (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using c and c++ programming languages. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment based on the processing language project. There are many types of Arduino in the market such as Arduino RS232, Arduino Diecimila, Arduino Uno R2, Arduino Leonardo.

In this project, Arduino mega is chosen as our main microcontroller

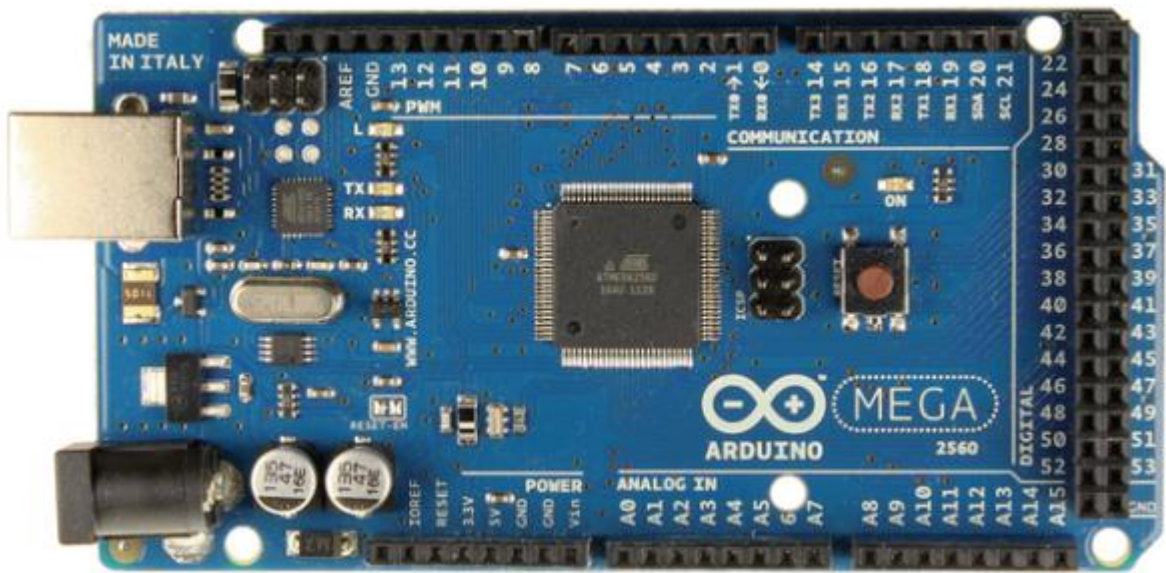


Figure 2 Arduino Mega – source: Internet

Example code:

Here is code that turns light in out repeatedly for one second.

```
LED_BUILTIN=11 // digital output D11

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

○ **Lcd - Liquid Crystal Display**

An electronic display device which has a flat piece of glass display which is commonly used in digital devices. In this project, LCD will display the readings and measurement of parameter. We have used 16x2 characters.



Figure 3 LCD display

SETUP/CONNECTION (LCD Display to Arduino)

The require materials for the project are listed below:

1. An Arduino board (Arduino Mega)
2. A breadboard
3. A potentiometer
4. Wires
5. An LCD Display

Table 1 Wiring Table LCD

LCD		Mega	
RS		Arduino Digital Pin 37	
E		Arduino Digital Pin 36	
D4		Arduino Digital Pin 35	
D5		Arduino Digital Pin 34	
D6		Arduino Digital Pin 33	
D7		Arduino Digital Pin 32	
D3	Rheostat middle leg	-	
D1	rheostat	Breadboard -ve power rail	Arduino GND
D2	rheostat	Breadboard +ve power rail	Arduino 5v

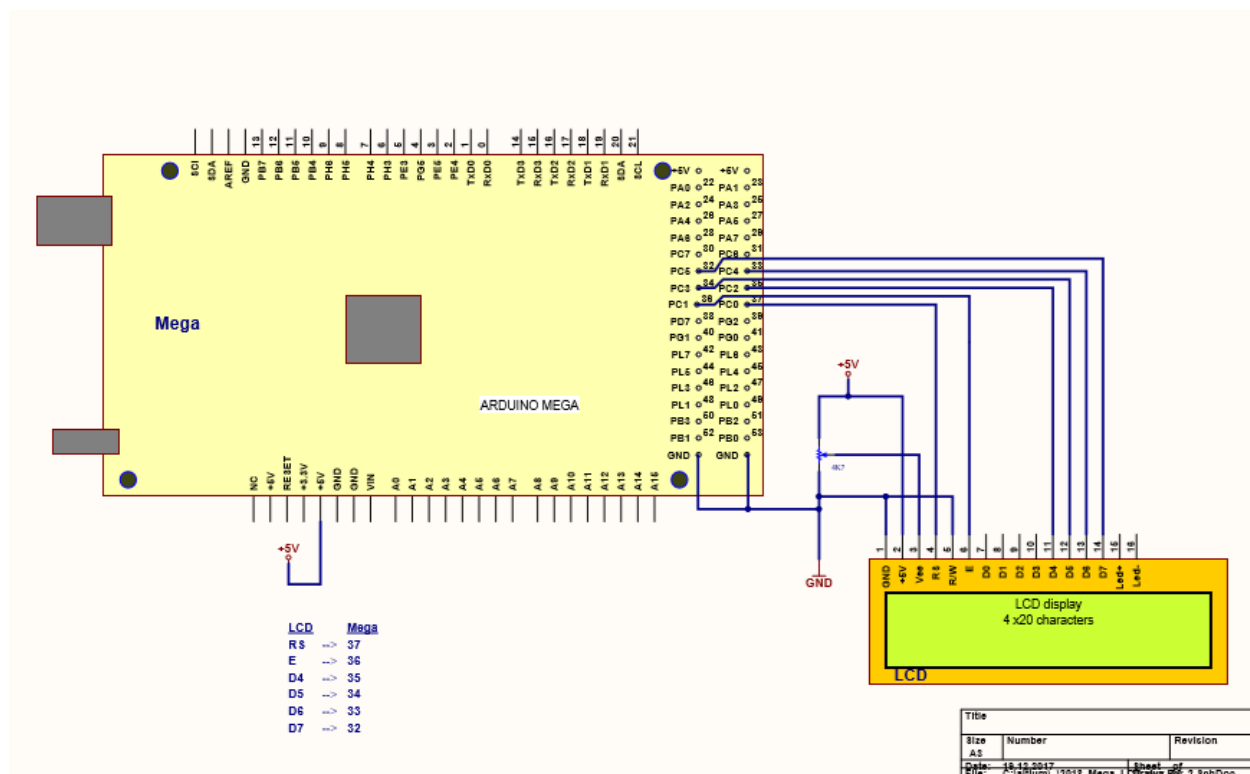
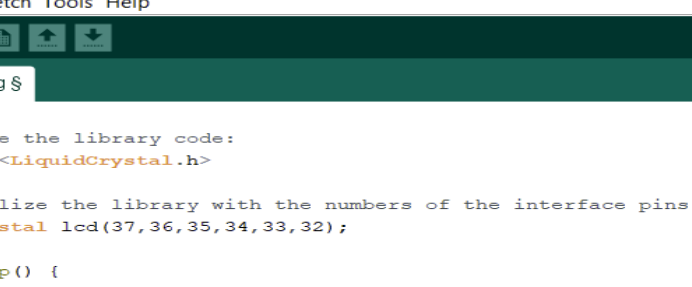


Figure 4 LCD diagram - source: Embedded Systems Course

Example Code



The screenshot shows the Arduino IDE interface. At the top, the title bar reads "lcd_testing | Arduino 1.8.12 (Windows Store 1.8.33.0)". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Under the "Sketch" menu, there are icons for "Verify", "Run", "New", "Open", and "Save". The main text area contains the following C++ code:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(37,36,35,34,33,32);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

At the bottom of the IDE, a status bar shows "Done Saving." and "Arduino Uno".

Figure 5 Code example LCD

- **Ethernet Module Arduino**

The Ethernet module used in the project is W5100, which is a full-featured, single-chip Internet-enabled 10/100 Ethernet controller designed for embedded applications where ease of integration, stability, performance, area and system cost control are required.

NB: Pins 10,11,12 and 13 are reserved for interfacing with the Ethernet module and should not be used otherwise. This reduces the number of available pins to 9, with 4 available as PWM outputs.

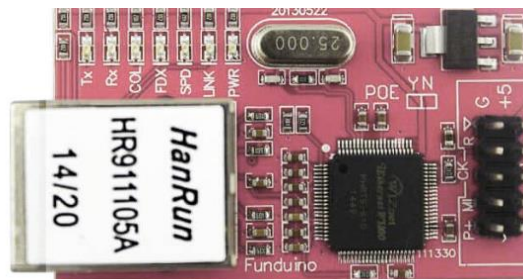


Figure 6 Ethernet module Arduino - source: Internet

SETUP/CONNECTION (Ethernet module to Arduino mega)

The required wiring and pins to connect the Ethernet module with Arduino is given below:

Table 2 Wiring Table Ethernet

Ethernet Module	Mega
GND	GND
+5v	+5v
MISO	50
MOSI	51
SS	10
SCK	52
RST	

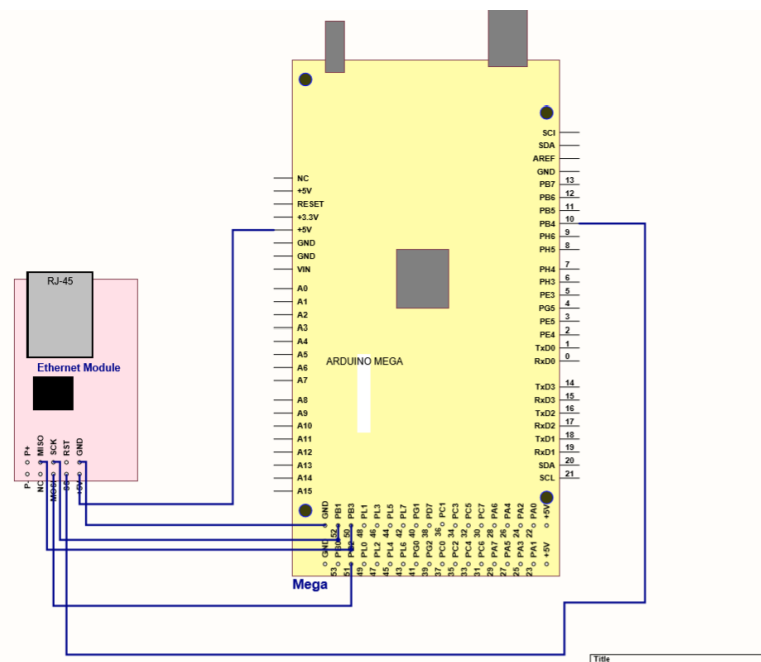


Figure 7: Wiring Ethernet module to Arduino mega – source: Embedded Systems Course

Example Code

Folder: Ethernet_1_emb_systems_MQTT_3_2020

- **MQTT CLIENT TO MQTT BROKER**

Once the board is finished and Ethernet has been tested, it is time to publish the data signal to the c++ gateway.

As described in the figure 8 below, while compiling the file *Ethernet_1_emb_systems_MQTT_3_2020.ino* turn on the terminal emulation software:

Windows: WinSCP, Putty
Mac: terminal application inside

Then logging into the VM (VPN opened):

Local IP: 172.16.200.88

Port: 22

Username: iotti

Pwd: iotti2017

- ⇒ Subscribe to the topic where all the local machine's data is sending to
- ⇒ `Mosquitto_sub -t "ICT4_out_2020"`
- ⇒ In here, it's possible to view all the latest weather JSON data sent to the MQTT broker in form of `IOTJS={"S_name": "name", "S_value": value}`

This project mainly focuses on the humidity signal

Therefore, the signal sending would be

- ⇒ `IOTJS={"S_name": "Humidity_out", "S_value": X}`

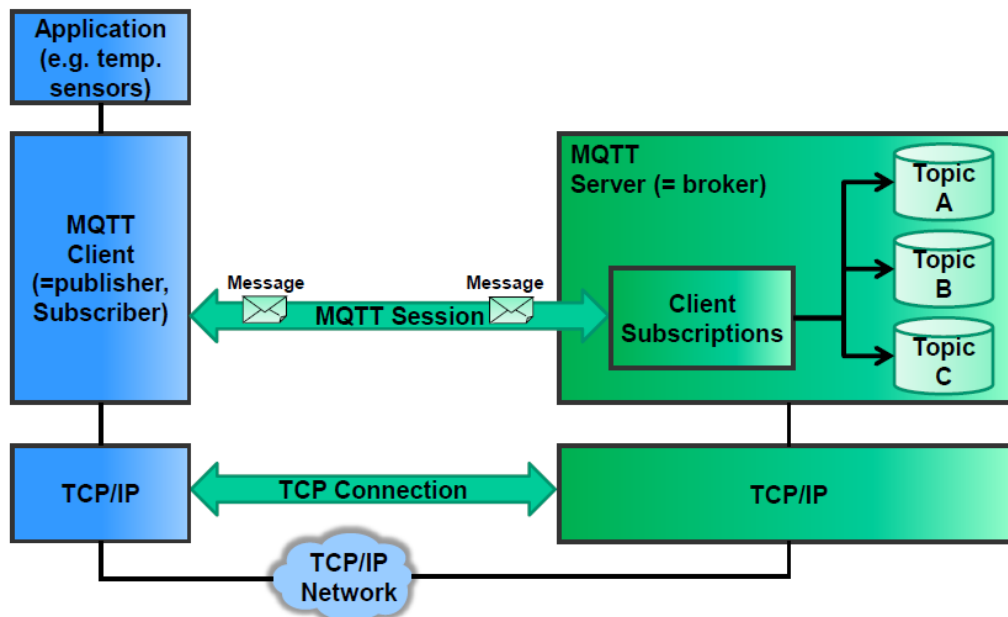
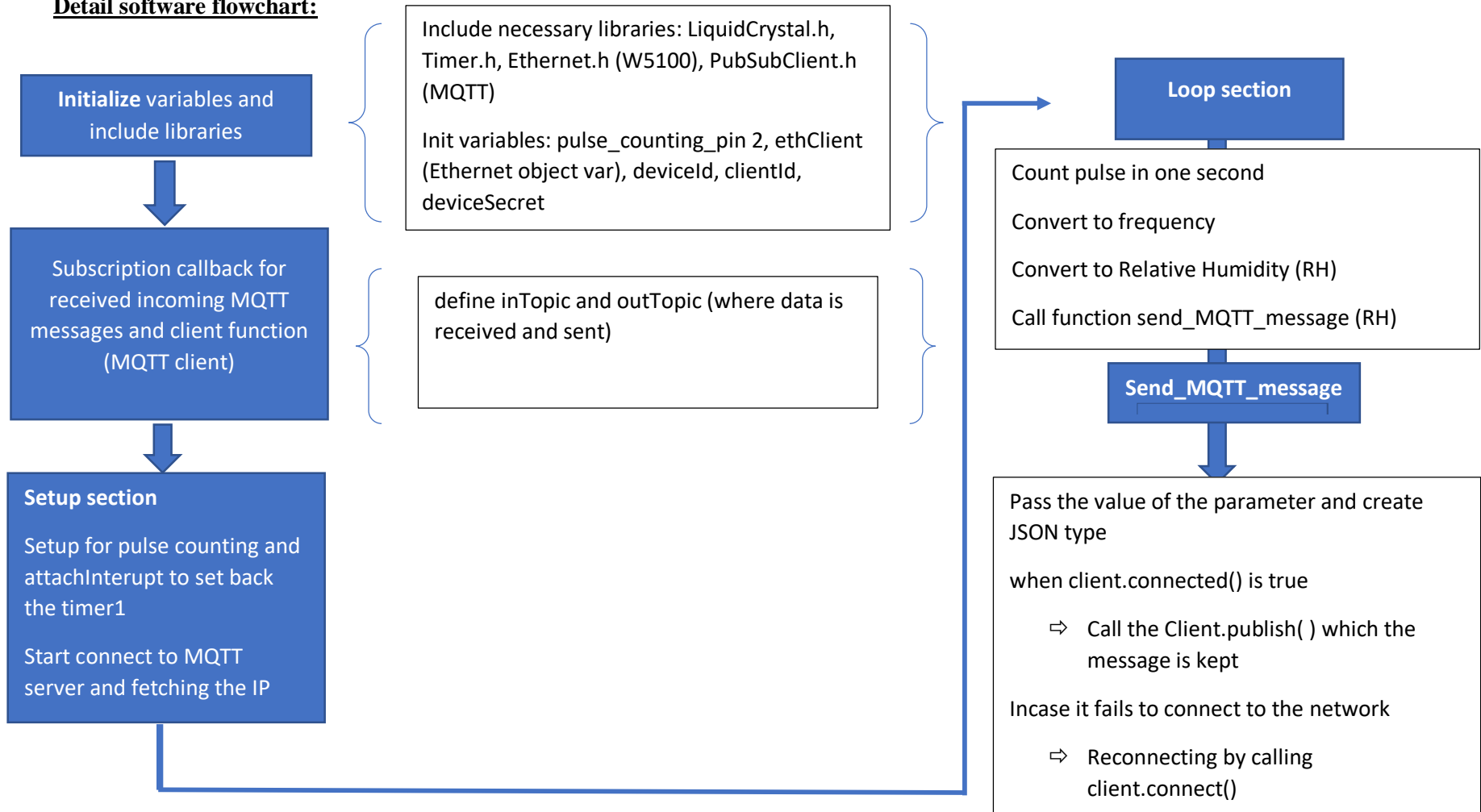


Figure 8 Flowchart from MQTT client to server – source: Embedded System Course

Detail software flowchart:



○ **MQTT SERVER TO REST API**

Once the data is transform and sending per second to the MQTT broker. It is time push the data to the database (<http://bowd12-api.course.tamk.cloud/v1/weather>)

Open terminal and log in VM with the username and password mentioned in the section **MQTT CLIENT TO MQTT BROKER**

Open directory Embsys:

- Start threads: main.cpp
- Mqtt_arduino1.cpp
- Send_tamk.cpp

The way software works is described in the figure 9

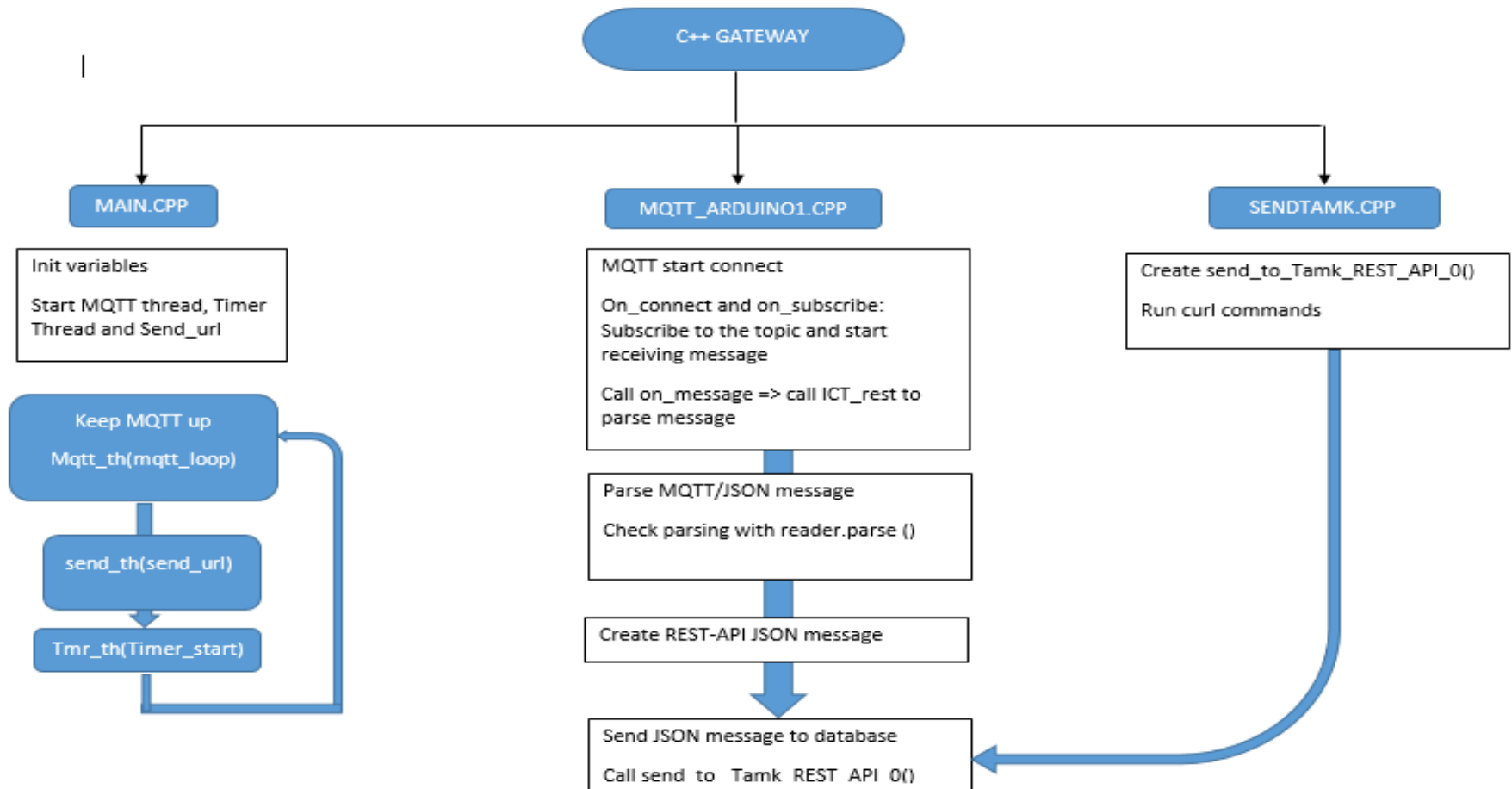


Figure 9 Software Flow chart of C++ gateway – source: Embedded System course

Main.cpp

Start MQTT threads includes:

- Mqtt_loop(checking if the messages is coming or not)
- Timer functions: to set time interval to send to database (sending rate = 10s)
- Send_url (to count the number of messages passed to DB)

Mqtt_arduino1.cpp

A class mqtt with public functions:

- Mqtt: to connect to MQTT server
- On_connect: to subscribe to topic
- On_subscribe: confirm successful connection to MQTT topic
- On_message: check message is receiving from client
- ICT_rest: the function to separate the message information, take S_name, S_value, S_path, S_unit, S_data => create the json format message and send it to database in here call the function send_to_Tamk_REST_API_0 () – called in the SendTamk.h

Send_tamk.cpp

Snd_to_Tamk_REST_API_0 () Send the message to the database under the right format with 2 parameters: api ip and string value of the JSONs message

5. REFERENCES

Project_Plan.docx

Test_Case.docx

Test_Plan.docx

Test_Report.docx

