



UPPSALA
UNIVERSITET

ISRN UTH-INGUTB-EX-E-2023/002-SE

Examensarbete 15 hp

Mars 2023

Home automation using the Internet of Things

Ronghan Wu



UPPSALA
UNIVERSITET

Home automation using the Internet of Things

Ronghan Wu

Abstract

This bachelor's thesis project is intended to study IoT (Internet of Things) and apply it to an application in home automation. To this purpose, a prototype has been implemented that can measure indoor temperature and humidity, display the measurement data on an LCD display, and then send the measurement data to a cloud server for users to read online. The hardware of the system consists of a digital temperature and humidity sensor (DHT11), a microcontroller board (Arduino MKR 1000 Wi-Fi) and an LCD display. The measurement is made with the sensor under the control by the microcontroller and shown on the LCD display; and at the same time, the measurement data is sent via Wi-Fi (available on the board) to the cloud server to display the data on a user interface on the cloud. The low power consumption and low cost as well as security have been taken into account in the implementation.

The prototype has been tested and evaluated, and it is shown to work well as above mentioned. Both Arduino IDE and Arduino IoT Cloud platform are strong development tools to extend the capability and performance for home automation with IoT. Future work can be done on developing a larger IoT network for home automation, a user-friendly interface for both PC and mobile devices, and an indicator showing energy consumption and battery life.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Uppsala/Visby

Handledare: Anthon Jonsson Ämnesgranskare: Ping Wu

Examinator: Andrej Savin

Contents

1	Introduction	1
1.1	Background	1
1.2	Objectives	1
1.3	Tasks and scope	1
2	Theory	2
2.1	Communication protocol	2
2.1.1	Application layer	2
2.1.2	Transport layer	3
2.1.3	Network layer	3
2.1.4	Link layer	4
2.1.5	Physical layer	4
2.2	Internet of Things	4
2.2.1	Enabling technologies	5
2.2.2	Wi-Fi	5
2.2.3	Wireless sensor network	6
3	Implementation	7
3.1	Overview of the system	7
3.2	Hardware components	8
3.2.1	Arduino MKR1000 Wi-Fi	8
3.2.2	LCD Display Module	9
3.2.3	DHT11 digital temperature and humidity sensor	10
3.3	Software and development tools	11
3.3.1	Arduino IDE	11
3.3.2	Arduino IoT Cloud	11
3.3.3	Google Drive	12
3.3.4	PushingBox	12
3.4	Assembly and Execution	13
3.4.1	Temperature & Humidity display	13

3.4.2 Google Sheets logging system	14
3.4.2.1 Google service	14
3.4.2.2 Configuring PushingBox.....	14
3.4.2.3 Software development	15
4 Result and Discussion	16
5 Conclusions and Further Work	18
5.1 Conclusion	18
5.2 Further development and improvements	18
References.....	19

Abbreviations

Acronym	Definition
API	Application programming interface
CoAP	Constrained application protocol
HTTP	Hypertext transfer protocol
IEEE	Institute of electrical and electronics engineers
IoT	Internet of Things
IP	Internet protocol
IPv4	Internet protocol version 4
IPv6	Internet protocol version 6
LLC	Logical link control
LLN	Low-power and lossy network
MAC	Medium access control
MQTT	Message queuing telemetry transport
OS	Operation system
TCP	Transmission control protocol
UDP	User datagram protocol
UI	User interface
WPAN	Wireless person area network
WSN	Wireless sensor network

1 Introduction

1.1 Background

In the recent years, home automation has become popular. It is building automation for home use, also called Smart Home or Smart House. Among the leading brands are, for instance, Google, Amazon, Apple and Xiaomi. A home automation system could control lighting, climate, entertainment systems, appliances, and even home security such as access control and alarm systems. When connected with the Internet, home devices are important constituent of IoT (Internet of Things). By using IoT, people would be able to monitor and control local wireless sensor-actuator networks through Internet, by using cloud and edge computing network anywhere in the world.

While there are many competing vendors, there are only few worldwide accepted industry standards and the smart home space is heavily fragmented. Each manufacture wants their own standard for customer stickiness and better profit. This leads to the idea of this project to learn and build a suitable home automation system in IoT direction, for users who are away from home, for example on vacation. In this case, it is usually two things bother people, home security and watering plants. The system will be able to monitor and configure the devices in the networks via user interface. And it needs to be easy to expand when needed.

1.2 Objectives

The objective is to develop a prototype for a simple, low-cost home automation system in IoT direction. That requires a low cost but sufficiently good performance. A cloud-based logging system to automatically capture important information has been taken. An important thing is that devices should be energy efficient to have long enough endurance.

1.3 Tasks and scope

“Home automation using Internet of Things” is a huge topic. This project is conducted only with the following specific tasks:

- Studying wireless protocols and IoT
- Building a local wireless sensor network
- Setting up for IoT use
- Connecting local wireless network to internet
- Making two-way communications between sensor nodes and end users
- Creating user interface for monitoring and controlling

2 Theory

2.1 Communication protocol

Communication protocol is a system of rules and conventions that allows different communications systems to transmit information via a variety of physical communication media, e.g., wired or wireless media. [1] The rules can be expressed by algorithms and data structures. Protocols are structured in layers to form a protocol stack in modern protocol design, that divides the task into smaller steps, each of which deals with a specific part, and interacting with other layers in a small number of well-defined ways. It keeps each part of a protocol to be relatively simple without a combinatorial explosion of cases.

The communication protocols in use on the Internet are called Internet Protocol Suite. Internet protocol defines the format and the order of messages exchanged between different communicating entities. It is divided into different layers, and each layer implements a service. The most common Internet protocol stack includes Application layer, Transport layer, Network layer, Link layer, and Physical layer.

The OSI model (Open Systems Interconnection model) is a conceptual model from ISO (International Organization for Standardization) based on experience with networks that predated the internet for a general communication with strict rules of protocol interaction and tight layering. It is a 7-layers model which are: Application, Presentation, Session, Transport, Network, Data Link, and Physical. The functions of each layer communicate and interact with the layers above and below it. [2]

Meanwhile, in the IoT area, there is no universally agreed architecture. The most common architecture is five layers: Application layer, Transport layer, Network layer, Link Layer and Physical layer, which are listed specifically in Table 2.1, and explained in detail in the following sub-sections.

Layer	Protocol
Application	HTTP
Transport	TCP/UDP
Network	IPv4/IPv6
Link	Ethernet
Physical	Bits “on the wire”

Table 2.1: A typical IoT internet protocol

2.1.1 Application layer

Application layer is the top layer in the 5-layer Internet protocol layer. One of the most common Application layer protocol people uses every day is Hypertext Transfer Protocol (HTTP). HTTP is a request-response protocol in the client-server computing model. The client submits an HTTP to the server, and server provides content, such as HTML files, or functions.

2.1.2 Transport layer

Transport layer is for process-process data transfer. It has TCP and UDP, two different kinds of main protocols. It provides logical communication between app processes running on different hosts. More than one transport protocol available to apps.

UDP

UDP, the User Datagram Protocol, is a simple but also core protocol in Transport layer. It is unreliable, has unordered delivery. No-frills extension of “best-effort” IP. No connection establishment, simple, small header size, no congestion control. UDP segments maybe lost and delivered out-of-order to app. UDP is suitable for applications such as streaming multimedia apps, IoT sensor data.

TCP

TCP, the Transmission Control Protocol, is another main protocol in Transport layer. Unlike UDP, TCP differs in several important features. TCP provides ordered data transfer, re-transmission, error-checked, flow control, congestion control. Delivery between applications running on hosts communicating via IP network. TCP is reliable, in-order delivery.

	UDP	TCP
Reliability	Unreliable	Reliable
Data sequencing	Able	Unable
Header size	Small	Large
Error checking	Extensive error checking and acknowledgment of data	Basic error checking mechanism using checksums
Speed	Faster	Slower

Table 2.2: Comparison of UDP and TCP protocols [3]

2.1.3 Network layer

Network Layer transports segment from sending to receiving host. The two key functions of Network layer are forwarding and routing: Moving packets from router’s input to appropriate output; Determining route taken by packets from source to destination. [4]

IP address (IPv4 and IPv6)

Internet Protocol address (IP address) is a numerical address to each device connected to network. It has two main functions, host or network interface identification, and location addressing.

Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol. It uses a 32-bit number address, which provides 4,294,967,296 (2^{32}) unique address. But dude to lots of addresses are reserved for special use, the IPv4 address has been exhausted at the IANA level since 2011. This leads to the next generation Internet Protocol version 6 (IPv6).

Internet Protocol version 6 (IPv6) is the most recent version of IP, and it was first introduced December 1995. It increased the address size to 128 bits and can provide 3.403×10^{38} (2^{128}) addresses, which is nearly infinite for the foreseeable future.

	IPv4	IPv6
Size of IP address	32-bit	128-bit
Addressing method	Numeric address, separated by dot (.)	Alphanumeric address, separated by colon (:)
Check sum	Yes	No
Packet size	576 bytes and optional fragmentation	1208 bytes without fragmentation
Security	Depending on applications	Internet Protocol Security is built in
Address configuration	Manual or via DHCP (Dynamic Host Configuration Protocol)	Autoconfiguration stateless address using ICMPv6 (Internet Control Message Protocol version 6) or DHCPv6
Mapping	ARP (Address Resolution Protocol) to map to MAC address	NDP (Neighbor Discovery Protocol) to map to MAC address

Table 2.3: Comparison of IPv4 and IPv6 protocols [5]

2.1.4 Link layer

Link layer provides a node-to-node data transfer. It detects and possibly corrects errors that may happen in the physical layer. IEEE 802 divides the link layer into two sublayers: Medium access control (MAC) layer, and Logical link control (LLC) Layer. Together with 802.3 Ethernet, 802.11 Wi-Fi, and 802.15.4 ZigBee, these are the most common in Link layer.

2.1.5 Physical layer

Physical layer is the first and lowest layer in a five-layer Internet protocol. This layer is where raw bits stream over a physical transmission medium.

2.2 Internet of Things

The Internet of Things (IoT) is physical objects or groups of such objects with technologies that connect and exchange data with different devices and systems, over the Internet or other communication networks. [6] Traditional fields such as embedded systems, control systems, wireless sensor networks, and automation (both home and building automation), they enable the Internet of Things collectively and independently. It is often called “smart home” in the consumer market. Home automation can be based on hubs or platforms that control devices and applications.

There are usually 2 kinds of IoT protocols, 3- and 5-layer architectures. Three-layer is the most basic one, but when talking about research on IoT, we use more layered architectures for finer aspects of Internet of Things.

2.2.1 Enabling technologies

There are many technologies and ways to enable IoT. To fulfill the requirements to communicate between devices, several wired and wireless technologies can do. Different accessible and range, such as Bluetooth, Zigbee, Wi-Fi, 5G, Ethernet. There are a variety of communication protocols used for IoT.

Different technologies have different features. Below shows some of the most popular communication technologies in IoT area:

	Frequency	Data Rate	Range	Power Usage	Cost
Bluetooth LE	2.4Ghz	1, 2, 3 Mbps	90 meters	Low	Low
Zigbee	2.4Ghz	250 kbps	90 meters	Low	Medium
Z-Wave	subGhz	40 kbps	30 meters	Low	Medium
2G/3G	Cellular Bands	10 Mbps	Several kilometers	High	High
WirelessHART	2.4 Ghz	250 kbps	90 meters	Medium	Medium
Wi-Fi	subGhz, 2.4Ghz, 5Ghz	0.1-54 Mbps	<90 meters	Medium	Low

Table 2.4: Comparison of technologies [7]

The one this project used is Wi-Fi based, due to low cost and easy access.

2.2.2 Wi-Fi

Wi-Fi is a technology for local area networking based on the IEEE 802.11 standard. [8] It is commonly used for digital devices to exchange data and information by radio waves, typically for short range. Wi-Fi is one of the most commonly used technology in the world, in home and smaller office networks to link devices to a wireless router to connect them to the Internet.

Devices must use a certain common Wi-Fi version. Through the years since the first generation of Wi-Fi was introduced in 1997, many different versions of Wi-Fi were adopted.

Generation	IEEE standard	Adopted	Maximum link rate (Mbit/s)	Radio frequency (GHz)
Wi-Fi 6E	802.11ax	2020	574 to 9608	6

Wi-Fi 6	802.11ax	2019	574 to 9608	2.4/5
Wi-Fi 5	802.11ac	2014	433 to 6933	5
Wi-Fi 4	802.11n	2008	72 to 600	2.4/5
Wi-Fi 3	802.11g	2003	6 to 54	2.4
Wi-Fi 2	802.11a	1999	6 to 54	5
Wi-Fi 1	802.11b	1999	1 to 11	2.4
Wi-Fi 0	802.11	1997	1 to 2	2.4

Table 2.5: Wi-Fi generations

2.2.3 Wireless sensor network

A wireless sensor network (WSN) is a group of sensors that collect and wirelessly transmit data to a central location. A WSN is useful in measuring data like temperature, humidity, sound, light, etc.

A WSN is built of many nodes, which are connected to sensors. Each node has several parts: micro controller, radio transceiver, power supply (usually battery).

The topology of a WSN can be different, depending on applications. From simple single-hop network to more advanced multi-hop wireless mesh network. A hop means the number of nodes that data packet needs to go through to destination. In single-hop network, the data packet only pass through one router to its destination. In multi-hop networks, data packet pass through multiple routers. The star network topology, which belongs to single-hop network. In this project, single-hop network is used.

3 Implementation

3.1 Overview of the system

The system consists with 3 main parts, as shown in Figure 3.1. Arduino microcontroller gathers and process temperature and humidity data from sensor. Sensor value can be accessed via local LCD display, and red LED lights up if values are not to be expected (Figure 3.2). Then, Arduino sends data to PushingBox API server by using a specific device ID. This device ID is the key to send data forwards. After that, PushingBox pushes data to Google Drive, to log data on Google Spreadsheets. User can access logging value through smartphone or pc from their Google account service.

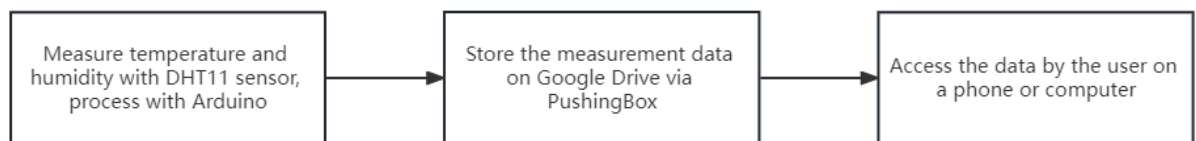


Figure 3.1: System overview

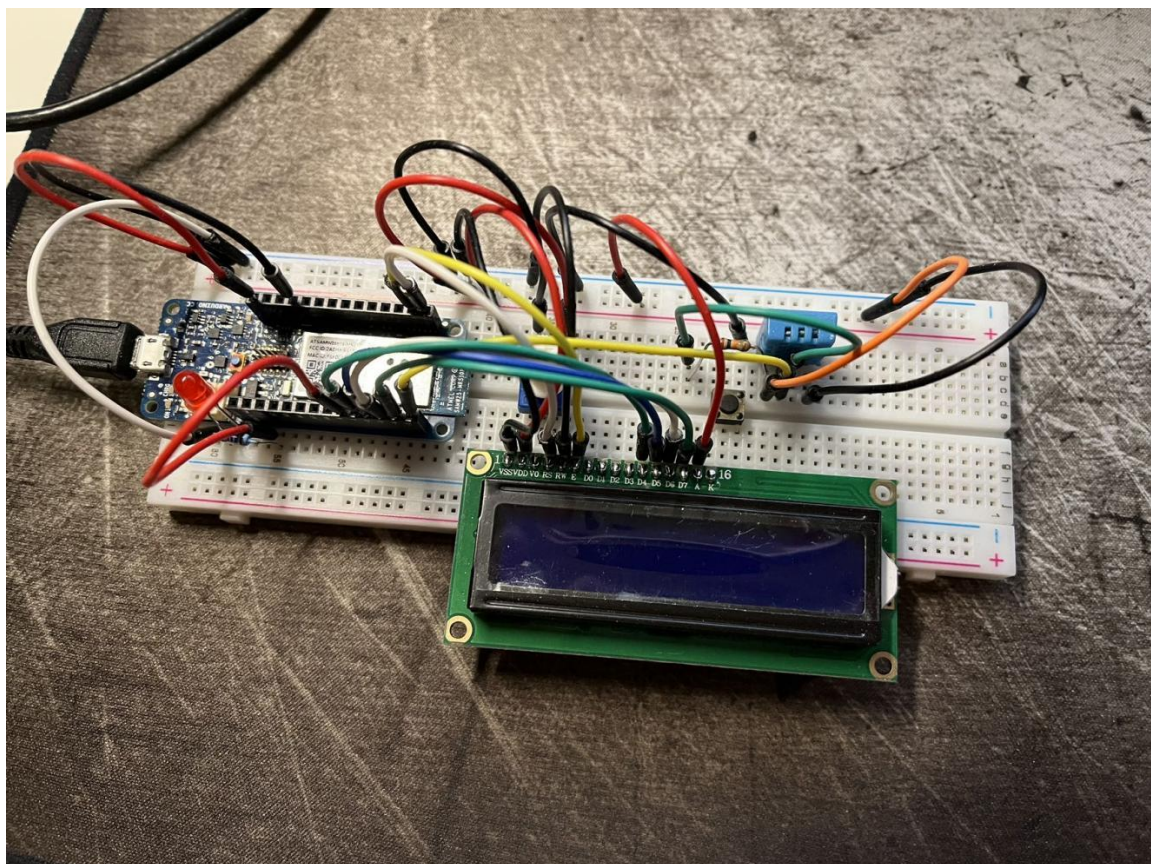


Figure 3.2: Hardware overview

3.2 Hardware components

3.2.1 Arduino MKR1000 Wi-Fi

Arduino MKR1000 Wi-Fi (Figure 3.3: (a) Arduino MKR1000 Wi-Fi board, and (b) its pin layout)) is a basic IoT and pico-network board that combines the functionality of the Arduino MKR Zero and the Wi-Fi Shield. It is powered by Arm Cortex-M0 32-bit SAMD21, a powerful and low-power processor. Also, ATSAMW25 Wi-Fi module focuses on power consumption and power saving mode. [9]

The technical specifications relevant to the project:

- Digital I/O Pins: 8
- Analog Input Pins: 7 (ADC 8/10/12 bit)
- Analog Output Pins: 1 (DAC 10 bit)
- Wi-Fi: ATWINC1500
- I/O Voltage: 3.3V
- Input Voltage (nominal): 5-5.5V
- Memory: 256KB Flash, 32KB SRAM

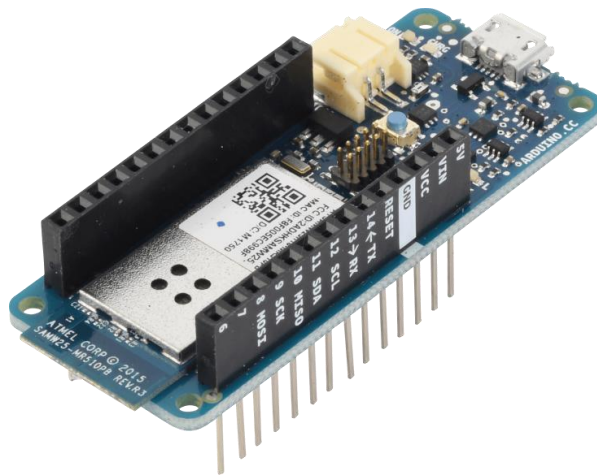


Figure 3.3: (a) Arduino MKR1000 Wi-Fi board

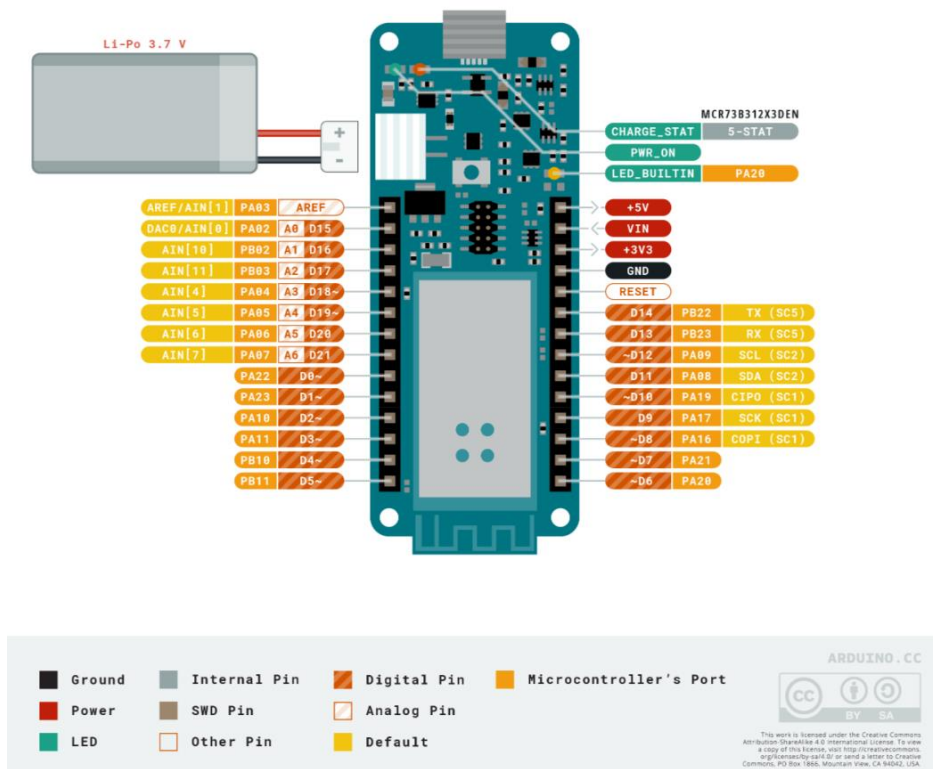


Figure 3.3: (b) MKR1000 Wi-Fi pin layout

3.2.2 LCD Display Module

HD44780 LCD Display (Figure 3.4) is driven by 5V and can display 2x16 characters. A 10k ohm potentiometer is used to adjust backlight strength for power saving.



Figure 3.4: LCD Display Module

3.2.3 DHT11 digital temperature and humidity sensor

DHT11 (Figure 3.5) is a basic, low-cost digital temperature and humidity sensor. It is based on a capacitive humidity sensor and a thermistor to measure and sends a digital signal on the data pin. It can measure humidity value for 0-100% with 2-5% accuracy, and temperature for -40 to 80 °C with ± 0.5 °C accuracy. 2.5 mA max current use during conversion (while requesting date). The only downside is that it has 0.5 Hz sampling rate, which means it can only capture data every 2 second. But for home use, it suits very well. [10]

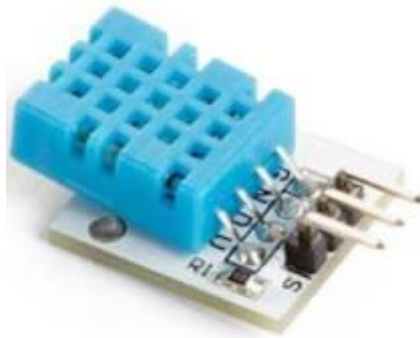


Figure 3.5: DHT11 temperature & humidity sensor

3.3 Software and development tools

3.3.1 Arduino IDE

Arduino IDE (Integrated Development Environment) (Figure 3.6) is Arduino official text editor for writing code and series monitor. It connects to the Arduino hardware to upload programs and communicate with them. [11]



Figure 3.6: Arduino IDE layout

3.3.2 Arduino IoT Cloud

Arduino IoT Cloud (Figure 3.7) is an application to make IoT project in a quick, easy and secure way. It can connect multiple devices to each other and allow them to exchange real-time data. It can also monitor them from internet by using available user interface, such as web browser and apps. Arduino IoT Cloud is also fully compatible with Arduino ecosystem. One can generate a template code in Arduino IoT Cloud and then edit and upload to board using the web editor. [12]



Figure 3.7: Arduino IoT Cloud widgets in a dashboard

3.3.3 Google Drive

Google Drive (Figure 3.8) is a file storage and synchronization service developed by Google. Google Drive allows users to store files in the cloud, synchronize across devices, and share files. Google Drive includes Google Docs, Google Sheets, and Google Slides, which are a part of the Google Docs Editors, permits collaborative editing of documents, spreadsheets, presentations, and so on. This project will focus on Google Sheets, to store sensor data in the cloud. [13]



Figure 3.8: Google Drive and its service

3.3.4 PushingBox

Pushing Box (Figure 3.9) is a cloud that can send in-real-time notifications based on API calls. Combine with Google Drive, it allows data to be palatable to Google Sheets. [14]

The need for using the PushingBox API is to turn HTTP transmitted data into Google Compliant HTTPS encrypted data.



Figure 3.9: PushingBox used with the Arduino

3.4 Assembly and Execution

3.4.1 Temperature & Humidity display

In this project, Arduino IDE is used to edit and upload code. It is viable within few changes, to convert code to Arduino IoT Cloud to be able to execute more function.

To start, plug the LCD in the breadboard and then connect it to the Arduino MKR 1000 board by the following schematic. Pin3 is connected with a 10k ohm potentiometer to adjust backlight strength level for power saving. Then connect the DHT11 sensor to ground and to +5V and to Arduino board. A red LED is connected to give an alarm light when room temperature is higher than expected.

To set up code, the DHT library from Adafruit is used. [15] By default, LCD display outputs Celsius degree, but can also change to Fahrenheit simply by modifying code.

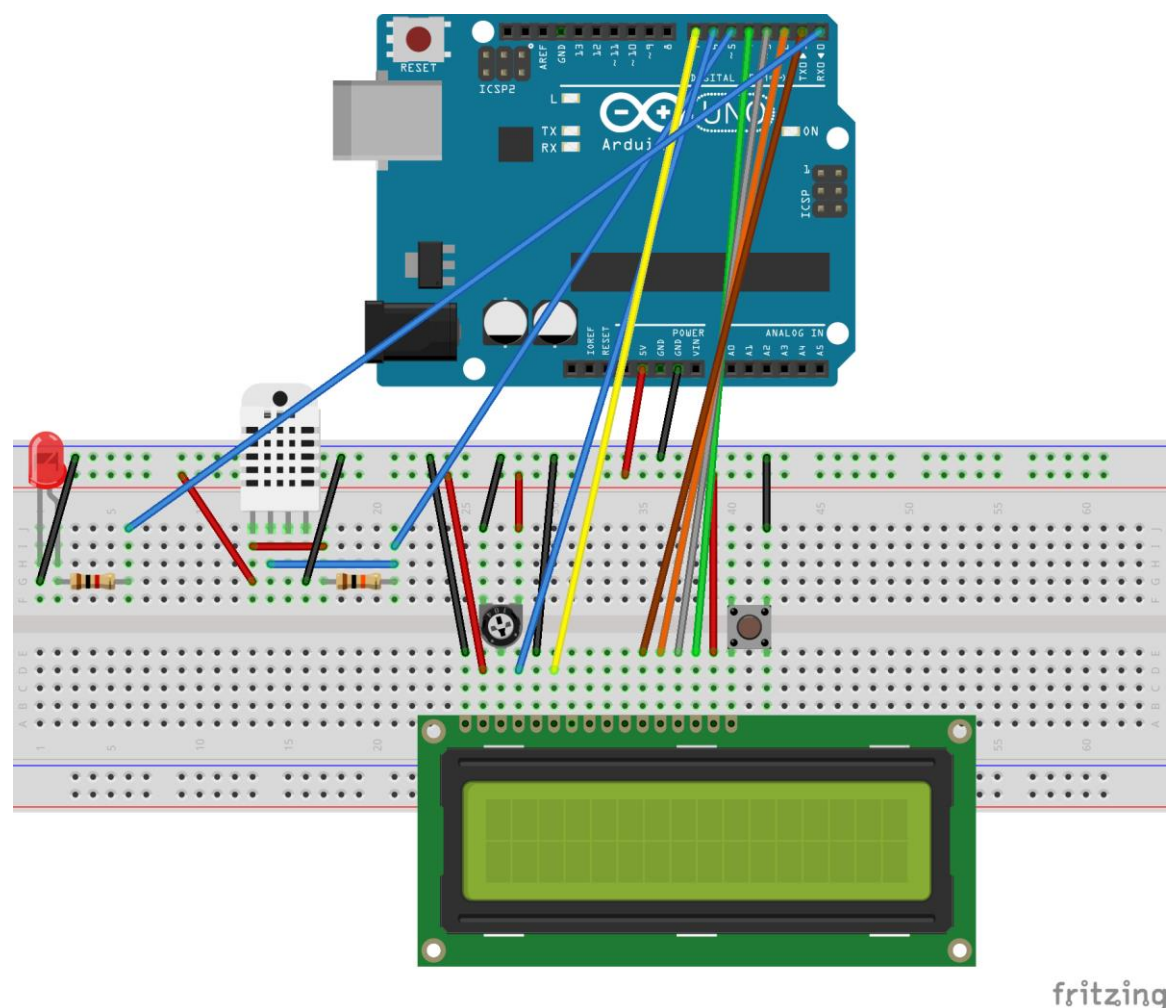


Figure 3.10: Fritzing schematic of entire system, UNO considering as MKR1000

3.4.2 Google Sheets logging system

3.4.2.1 Google service

To begin, there are a few steps need to be done.

Signing up a Google account. Then create a new Google Sheets. This spreadsheet will be populated by DHT sensor values by PushingBox. The device ID (URL key) needs to be copied and saved, that's the key in the URL between the “/d” and the “/edit” of the spreadsheet (Figure 3.11).

```
17
18 const char WEBSITE[] = "api.pushingbox.com"; //pushingbox API server
19 const String devid = "v48C7C601A6A0DF8"; //device ID on Pushingbox
20
21 const char* MY_SSID = "Betty";
22 const char* MY_PWD = "welcome1234";
23
24
25 int status = WL_IDLE_STATUS;
```

Figure 3.11: Device ID (URL key) and Wi-Fi setup

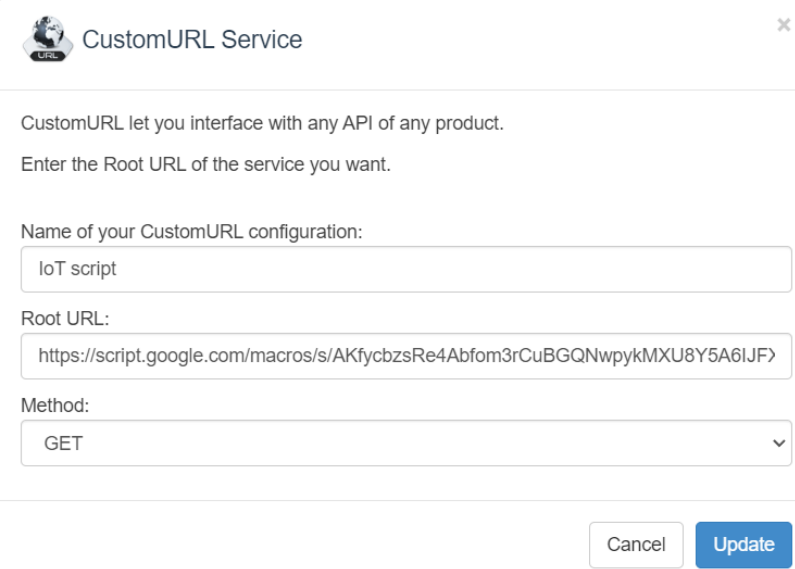
After that, create a Google App Script which will process sensor data and populate the spreadsheets. The device ID saved previously needs to paste into the correct line in the Google Script code. Then publish, deploy as a web app.

3.4.2.2 Configuring PushingBox

In the PushingBox, select “Add a Service” and “CustomURL Service”. Root URL would be Google App Script address saved in the first step. Method chooses “GET”. (Figure 3.12 (a))

After that, creating a scenario for the service. This can be done via “My Scenario”. Using “GET” method, and list all the sensor data variables need to be posted. Here we want “humidityData”, “celData”, “fehrData”, “hicData”, and “hifData”, statement begins with “?” to indicate the method we use is “GET”.

By that, PushingBox can receive sensor value sending from Arduino via Wi-Fi, and then pushes forwards to Google Drive.



CustomURL Service

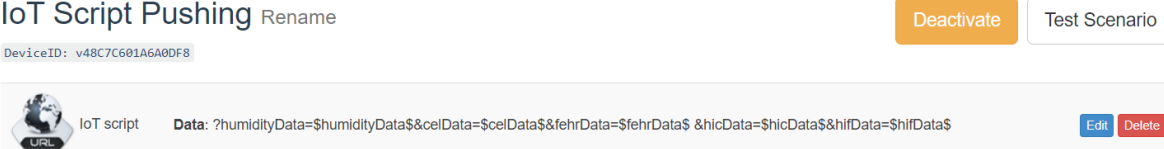
CustomURL let you interface with any API of any product.
Enter the Root URL of the service you want.

Name of your CustomURL configuration:

Root URL:

Method:

Figure 3.12: (a) CustomURL Service settings



IoT Script Pushing [Rename](#)

DeviceID: v48C7C601A6A0DF8


 IoT script **Data:** ?humidityData=\$humidityData&celData=\$celData&fehrData=\$fehrData\$ &hicData=\$hicData\$&hifData=\$hifData\$

Figure 3.12: (b) Scenarios of pushing data to Google Sheets

3.4.2.3 Software development

An addition Arduino library WiFi101 needs to download and update MRK 1000 board with newest firmware. [16] This is essential otherwise some boards with older firmware may not read data correctly.

In the coding part, “SPI” library is used, this allows us to set up Wi-Fi connection settings and also PushingBox API server. Because of free version PushingBox can only send 100 notifications a day, so we want to make the log interval longer enough to cover a whole day, that would be 15 minutes (90000ms) per logging. “client.connect()” and “client.stop();” need to be added to make sure connection gate can open properly in each data sending to PushingBox.

4 Result and Discussion

The development of the system in this project resulted in an accuracy sensor data output and cloud logging system for IoT purposes.

The system consists of a sensor node with Wi-Fi connection ability. The sensor node in this project is built by Arduino MKR1000 board, DHT11 temperature & humidity sensor, and an LCD monitor. Temperature and humidity data are captured and shown on the LCD monitor. User can get notification when a red LED lights up if temperature is higher than required temperature. (Figure 4.1) Sensor data is sent to PushingBox API via Wi-Fi, and converted to Google Drive compatible form to auto log every 15 minutes. It is viable to check connection information and data value through Arduino IDE locally, mostly for testing and maintain purpose.

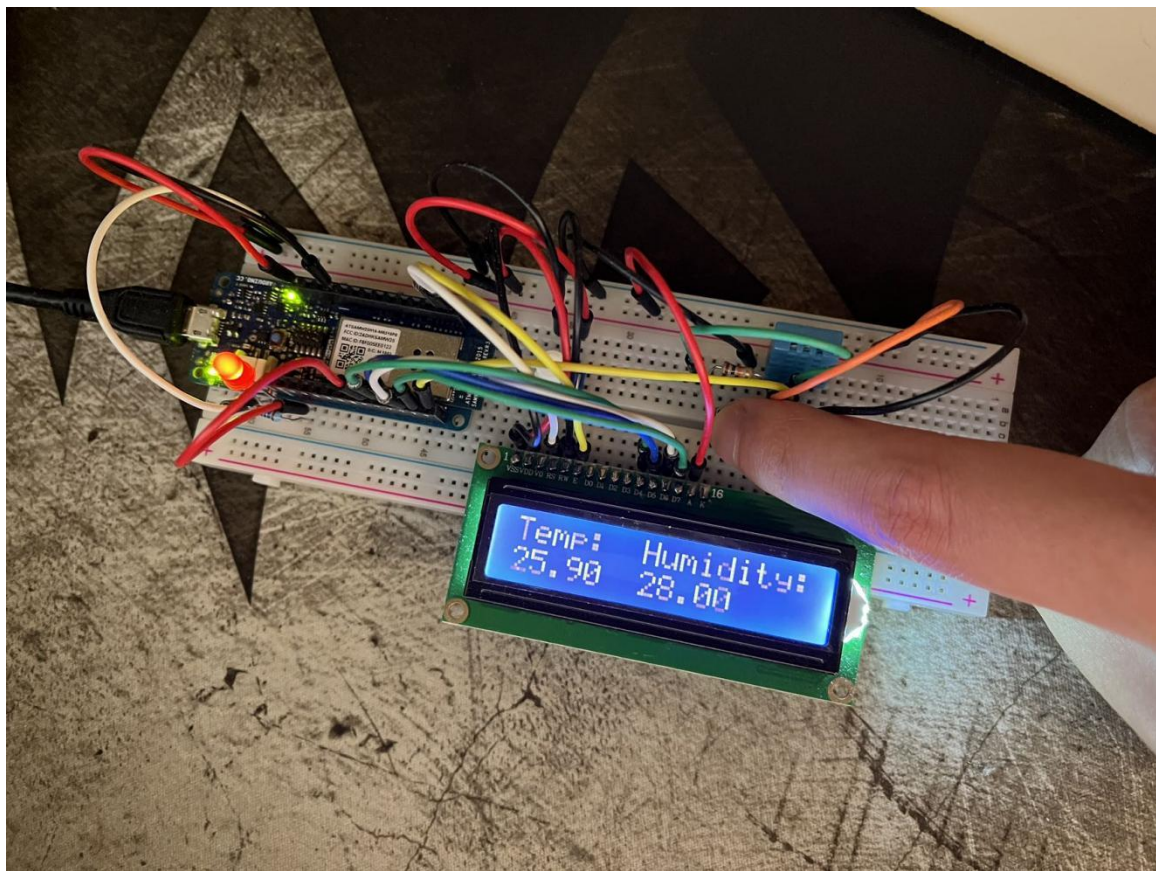


Figure 4.1: Finished running project

Arduino coding and internet connection knowledge has been studied. By testing and comparing with Arduino IoT Cloud, Arduino IDE can do most of the work and can convert code to Arduino IoT Cloud if further bidirectional interaction is needed. Due to the limited number of pins on Arduino MKR1000 board and many of the pins used for the LCD monitor, it can't connect more devices for more functions.

PushingBox with Google Drive can easily set up a connection to record data. Due to free version of service can only send 100 pushes each day, logging interval has to be at every 15 minutes, which is not optimal. Within this project goal, it is sufficient for a prototype. (Figure 4.2)

	A	B	C	D	E	F	
1	Timestamp	Untitled Question					
2	Date	Humidity %	Celsius C	Fahrenheit F	Heat Index C	Heat Index F	
3	07/12/2022	21	26.5	79.7	25	78	
4	07/12/2022	20	25.7	78.26	24	76	
5	07/12/2022	17	25.7	78.26	24	76	
6	07/12/2022	17	25.5	77.9	24	76	
7	07/12/2022	19	25.7	78.26	24	76	
8	07/12/2022	27	25.7	78.26	25	77	
9	07/12/2022	28	25.9	78.62	25	77	
10	07/12/2022	28	26.1	78.98	25	77	
11	07/12/2022	30	26.4	79.52	25	78	
12							
13							
14							
15							
16							

Figure 4.2: Sensor data logging on Google Sheets

This single node can be powered by battery for mobile use. Power consumption is not tested, and can only make an estimation on paper.

5 Conclusions and Further Work

5.1 Conclusion

The goal with this project was originally developing a wireless sensor network system with IoT purposes.

The finished system is a single sensor network node which can communicate with cloud service through Wi-Fi protocol. The system is built around Arduino MKR1000 board, with help of PushingBox and Google Drive, together to perform an environment alarming and logging system. The system performs well with a reasonable Wi-Fi range for home use, according to MRK1000 Wi-Fi specs.

The system was about to use Zolertia RE-mote, together with Contiki and Ubidots. Unfortunately, the method was no longer fully supported. Which is the reason why Arduino was chosen to implement this project.

5.2 Further development and improvements

During the process of this project, some additional ideas and scopes came up for further development. Due to the time limitation, they will be put for future improvements and investigation. Some of the thoughts have been pointed out in section 4 Result and Discussion. These are:

- (1) Work with other Arduino board to execute the similar IoT purposes use. Due to the imitated number of digital input/output pins on MKR 1000 (most pins are used for LCD monitor), there are limited hardware to connect to make more functions in a single node. Arduino Uno with Wi-Fi module is a common combination for simple home automation project for example. They can be even lower cost, and with better compatibility for more existed project to convert to IoT purposes.
- (2) Converting project to Arduino IoT Cloud. During pre-study and testing, Arduino IoT Cloud has been found a power tool for home automation with IoT purpose. It has an easy and clean interface for user to be able to interact with their devices. It is compatible with existed Arduino codes, with minor changes. It is viable to set the project in Arduino IoT Cloud, and changing required danger temperature via Arduino IoT Cloud phone APP, for example.
- (3) Investigate the possibility to find a more energy efficient way to implement current project purpose. Power consumption during the system running has not been tested. This is important for user to know how long a battery can last while using portably.
- (4) Due to limitation of PushingBox, only 100 notifications can be sent to Google Drive for logging. Some other API service can be chosen for better possibility to record data more frequently and easily.

References

- [1] "Communication protocol," [Online]. Available: https://en.wikipedia.org/wiki/Communication_protocol. [Accessed 15 May 2023].
- [2] "OSI model," [Online]. Available: https://en.wikipedia.org/wiki/OSI_model. [Accessed 15 May 2023].
- [3] M. Cook, "TCP vs. UDP: What's the Difference?," 24 Oct 2017. [Online]. Available: <https://www.lifesize.com/blog/tcp-vs-udp/#:~:text=TCP%20is%20a%20connection%2Doriented,is%20only%20possible%20with%20TCP>. [Accessed 15 Apr 2023].
- [4] "Internet Protocol," [Online]. Available: https://en.wikipedia.org/wiki/Internet_Protocol. [Accessed 15 May 2023].
- [5] L. Williams, "IPv4 vs IPv6 – Difference Between Them," 1 Apr 2023. [Online]. Available: <https://www.guru99.com/difference-ipv4-vs-ipv6.html>. [Accessed 15 Apr 2023].
- [6] "Internet of things," [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things. [Accessed 15 May 2023].
- [7] "IoT Standards and Protocols," [Online]. Available: <https://www.postscapes.com/internet-of-things-protocols/>. [Accessed 13 Jun 2023].
- [8] "Wi-Fi," [Online]. Available: <https://en.wikipedia.org/wiki/Wi-Fi>. [Accessed 15 May 2023].
- [9] "MKR 1000 WiFi," Arduino, [Online]. Available: <https://docs.arduino.cc/hardware/mkr-1000-wifi>. [Accessed 15 Apr 2023].
- [10] "DHT11–Temperature and Humidity Sensor," Components101, 16 Jul 2021. [Online]. Available: <https://components101.com/sensors/dht11-temperature-sensor>. [Accessed 15 Apr 2023].
- [11] "Arduino Integrated Development Environment (IDE) v1," Arduino, [Online]. Available: <https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics>. [Accessed 15 Apr 2023].
- [12] "Arduino Cloud," Arduino, [Online]. Available: <https://cloud.arduino.cc/how-it-works/>. [Accessed 15 Apr 2023].
- [13] "Google Drive," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Google_Drive. [Accessed 15 Apr 2023].
- [14] "Pushingbox," [Online]. Available: <https://www.pushingbox.com/index.php>. [Accessed 15 Apr 2023].
- [15] "DHT-sensor-library," [Online]. Available: <https://github.com/adafruit/DHT-sensor-library>. [Accessed 15 Apr 2023].

[16] "WiFi101," [Online]. Available: <https://github.com/arduino-libraries/WiFi101>. [Accessed 15 Apr 2023].