CSE237 Final Project Report

Wireless E-paper Room Sign & Door Locking System

Xiaonan Shen
xis079@eng.ucsd.edu
A13811845

Yijun Yan
y4yan@eng.ucsd.edu
A53298401

March 26, 2020

1 Introduction & Motivation

In this project, our team made a digital room sign system showing the room schedule on an e-paper display and controlling the lock based on RFID identification. The hardware components including an RFID reader, a wireless module (nRF24L01+), an SG90 servo motor, and an e-paper display module connecting with an Atmega328p chip. An Raspberry Pi acts as a bridge between the 328 chip and the Internet. Also, we uses an AWS server to communicate with the Google Calendar API and send the data back to the RPi. We denote the 328 side as *device*, the RPi side as *host*, and the AWS server as *server*.

The main motivation of this project is to make the schedule management of conference and study rooms (like the ones in the CSE MS Commons) easier. We want to build a truly wireless room sign that can be mounted anywhere but still have the ability to show real-time information. Ideally, such a light system can be easily embedded onto any door and the lock is controlled by a single servo motor. The system is expected to be with low power consumption and low delay on both the host and the device (though we will focus on the device power consumption), and has a relatively cheap cost.

2 Related Work

There are several commercial e-paper based room sign products like the ones from Visix[1] and Visionect[2]. Also, there are plenty of commercial or open-sourced RFID based access control systems. Our goal is to build a device that put them together into the same system and do it power efficiently. The biggest advantage of our system is that we can deal with two highly related tasks (schedule display and access control) in a single system instead of two so that it will be cheaper and easier to maintain. Also, the fact of lower power consumption gives it the ability to run on battery or solar cells so we don't need the power cables hanging around. Arduino provides a tutorial[3] on how to use the ATmega328P chip independently as an Arduino Uno so that we can get rid of the things we don't need like the USB to TTY adapter, external crystal, and LEDs to further decrease the power consumption.

We are going to use the RF24 library for the wireless module, and the demo code from the manufacture for the e-paper display.

[1] `https://www.visix.com/products/meeting-room-signs/electronic-paper-room-signs/`
[2] `https://getjoan.com/`
[3] `https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard`

3 Hardware

3.1 Components and Setup

Table 1 lists the components we used in this project. Figure 1 shows the circuit. Basically, we have the RFID reader, the transceiver, and the e-Paper module shares the SPI pins. Each of the other pins on the peripheral devices uses a dedicated pin. Also, the IRQ pins of the RFID reader and the transceiver are not used on the device. A capacitor of 0.1 µF is added between the power pins of the transceiver on the client and a breakout adapter is used on the host side to remove the noise from the power supply. We uses a 10k Ohm pullup resistor on pin 1 (reset) of the Atmega328 chip to avoid unexpected restarts and we added a bush button as a reset button. We follows a tutorial[4] from Arduino website to burn the Arduino bootloader onto the 328 chip so that we can write and upload codes just like a normal Arduino Uno. A 3.3V voltage regulator is used because the servo motor has to be run on 5 V and everything else uses 3.3 V.

Type	Model	Manufacture
Board	Arduino Uno Rev3	Arduino AG
Board	Raspberry Pi 4	Raspberry Pi Foundation
Servo Motor	SG90 9G Servo Motor Kit	Miuzei
MCU	Atmega328p-pu	Atmel
Transciver	nRF24L01+	Makerfire
Adapter	nRF24L01+ Breakout Adapter	Makerfire
E-Paper Display	4.2inch e-Paper Module (B)	Waveshare
RFID Reader	Mifare RC522 RFID Reader Module	IZOKEE
Voltage Regulator	LD1117V33	STMicroelectronics

Table 1: Main components

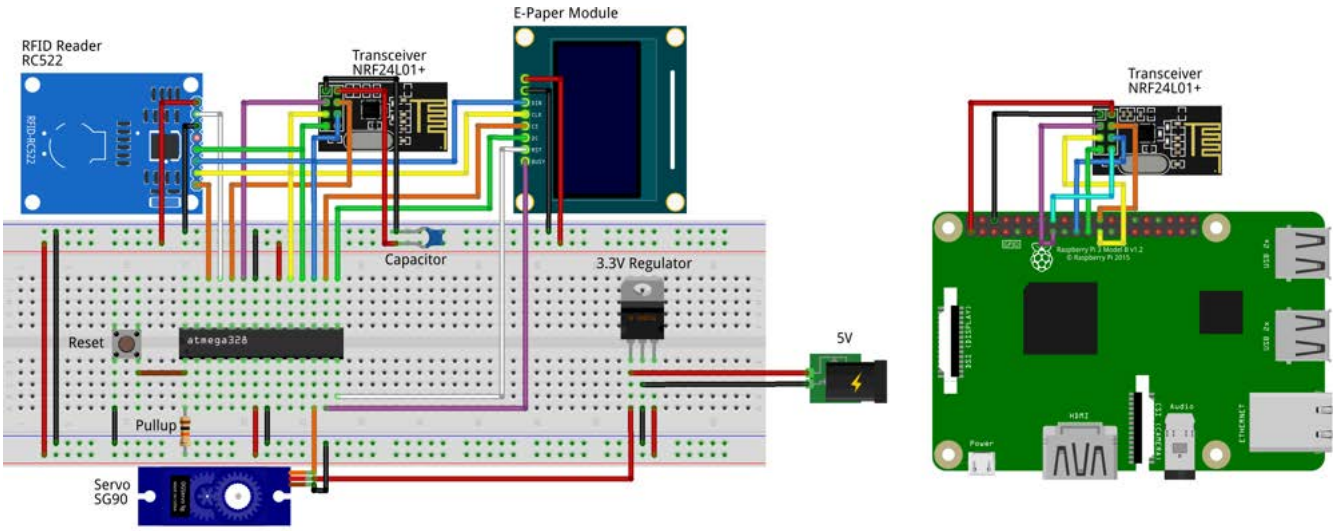

Figure 1: Circuit diagram of the device (left) and the host (right)

[4] https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard

3.2 Hardware Design Choices

In this project, we use a servo motor to represent the door lock system. Actually, it will be more reasonable to use electric strikes or electromagnetic locks for access control of conference and study rooms. However, those locks usually runs on 12V DC so we will need to do voltage shift and possibly use a relay. To make it simple and easier to compute the power consumption, we decided to just use a servo motor. Besides, we use a Raspberry Pi as the host. The host is actually doing very little job and acts only as the bridge between the device and the server. It is a workload that can be done using an Arduino. We are using a RPi for two reasons. First, each of us already had an RPi so we don't need to buy an extra device. And second, we are going to use WebSocket between the host and the server. It is a lot of pain to write WebSocket communication in C++ and make it work on an Arduino.

4 Software

4.1 Libraries

Table 2 lists the libraries we are using.

Library	Version	Description
AWS Server (Python)		
google-api-python-client	1.7.11	Communication with Google Calendar
AIOHTTP	3.6.2	POST requests and WebSocket connections
Raspberry Pi Host (Python)		
AIOHTTP	3.6.2	WebSocket connections
pyRF24	1.3.4	Communication with nRF24L01+ module
RPi.GPIO	0.7.0	Interrupt requests
Atmega328p-based Device (C++)		
Low-Power	1.6.0	Sleep
RF24	1.3.4	Communication with nRF24L01+ module
MFRC522	1.4.6	Communication with RC522 RFID reader
e-Paper	commit 8973995[5]	Communication with e-Paper display

Table 2: Software libraries

4.2 Software Design Choices

There is no major software design choice related to the libraries we use. Most of the libraries are basically the go-to choices. However, we did adjust the design of the system because of the software part of this project. The server is not in the original proposal of our project. We added it into the system because the Google Calendar API uses POST requests to do update notifications. Thus, we will either need to expose the RPi to the Internet or use a server. Exposing an RPi to the Internet requires setting up port forwarding on the router and that is not possible if we want to use it in various networks. Therefore, we ended up running a simple Python script on a server to deal with the bi-directional communication with the Google Calendar API.

We did make some software design choices that is related to the integration of the software and the hardware and that will be discussed in the next section.

[5] `https://github.com/waveshare/e-Paper/commit/8973995e53cb78bac6d1f8a66c2d398c18392f71` This is the demo code provided by the manufacture (Waveshare) and is not released as a library. Thus, it does not have a version number.

5 Integration

5.1 Overview

Figure 2 shows the communication between different parts of the system. When the host starts up, it establishes a WebSocket connection with the server. Then the server will fetch a list of upcoming events from the Google Calendar API and send it back to the Host. When there is an update, Google Calendar will make a POST request to the server and the server will fetch the updated list from Google Calendar and send it to the host through WebSocket. Here, to keep the script on the server simple, we did not have any status or data stored on the server side. It only acts as a bridge between the host and the Google Calendar by translating between WebSocket and HTTP requests.

Figure 3 is the flow char of the device. It checks for RFID tags every second and checks update of the event information every 10 seconds. Also, it put the chip and both modules into sleep mode as much as possible to save energy.

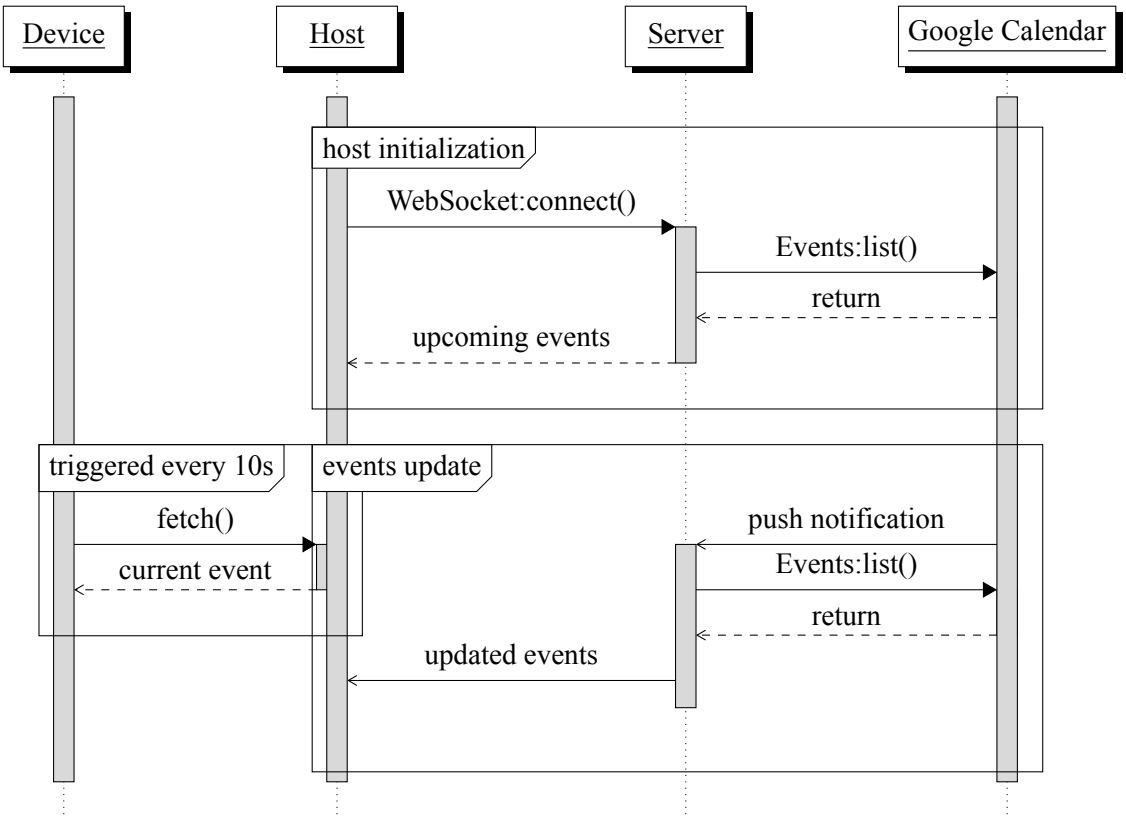

Figure 2: Sequence diagram of communication between different parts of the system

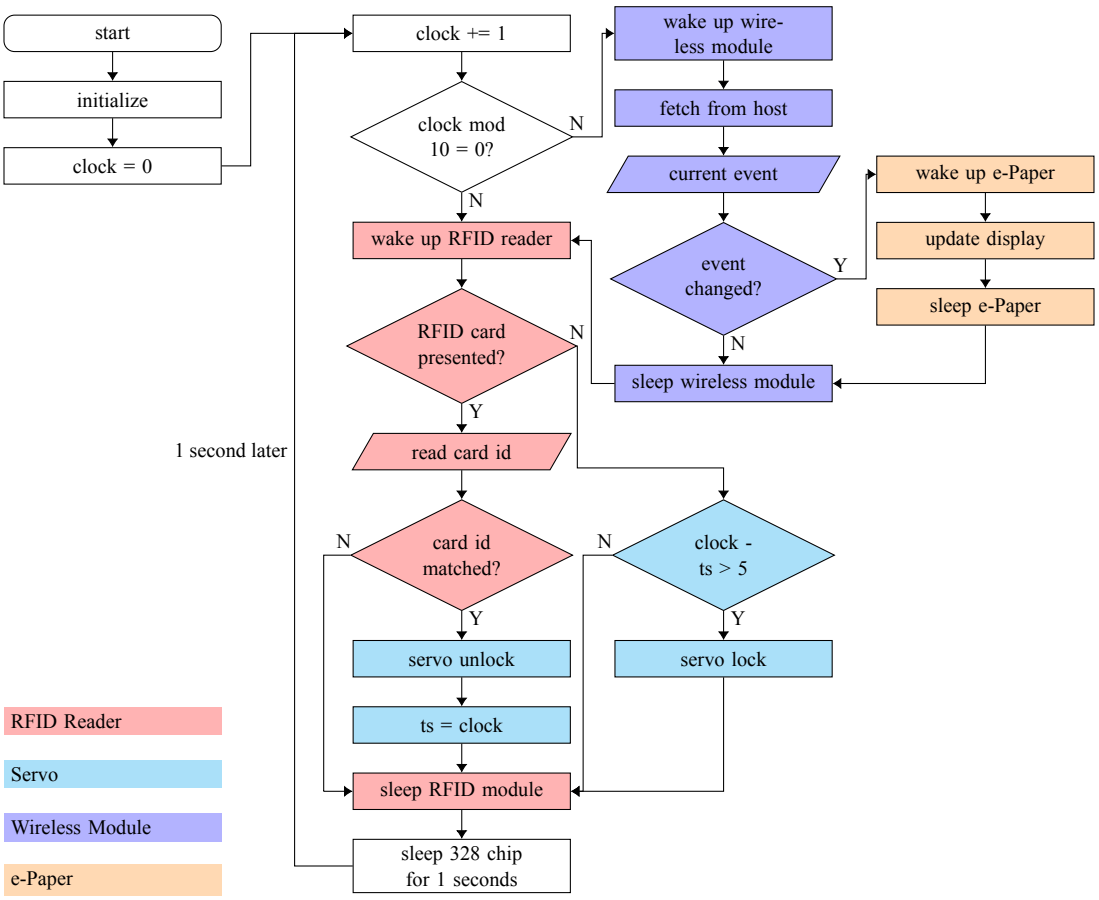

Figure 3: Flow chart of the software on the device

5.2 Challenges

Replacing Arduino Uno with Atmega328 chip significantly reduces the power consumption. However, the peripheral modules still consume a lot of energy. We've tried interrupts from the wireless module but it turns out that leave the nRF24 module in receiving mode consumes a lot of power (over 10mA at 3.3V). Therefore, we finally decided to use polling at an interval of 10 seconds. A 10-second delay is acceptable for room signs, and the update interval can be easily adjusted depending on either you want lower power consumption or shorter update delays. Besides, the RFID reader requires commands from the chip to detect tags. There is no way that the module can detect tags on its own and interrupt the chip. Therefore, the chip has to be waken up frequently (we choose once per second) to detect RFID tags, which makes the decision of using polling on the wireless module more reasonable.

Another challenge is that the nRF24 module can send at most 32 bytes of data in each package. We need to send a lot of data from the host to the device. First we tried to send the data of each event in multiple packages. It works well on the Arduino but as we move from Arduino to the 328 chip, since we reduce the frequency of the chip from 16MHz to 8MHz to get rid of the external crystal, the data sending process and auto ACK of the chip begins to fail. One big reason is that the host sends the next peace of data after it receives the ACK package from the device. However, the ACK package is automatically sent by the nRF24 module and when the frequency of the MCU is low, reading data out from the buffer on the module is much slower so the next peace of data may arrive when the buffer is still full, which causes failure. We can still make it work by manually sending ACK packages or adding delays on the host. However, the logic became very complicated and the debug process is painful. Therefore, we designed a new packet structure to fit the whole data into 32 bytes showing in Figure 4. This solution is not perfect. First, we use only 6 bits to represent a character so now the title and creator name only supports letters, numbers, and space.

Also, it is not scalable, so it is nearly impossible to add more information. Therefore, if we were to continue work on this project, we might switch back to the multi-package solution or try to use libraries like RF24Network that handles those things for us.

1 bit	96 bits	11 bits	11 bits	102 bits	32 bits	3 bits
available	title	start time	end time	creator	RFID key id	not used

Figure 4: 256 bits (32 bytes) packet structure of the data sent from the host to the device. Here, title and creator uses 6 bits to represent a character (only a-z, A-Z, 0-9, space, and \0). Start and end time are in the form of minute of day.

Besides, the screen has a resolution of 400x300 and three different colors, so it requires $400/8 \times 2 \approx 30KB$ memory to store a single frame of the image. However, the 328 chip has only 2 KB dynamic memory. Therefore, we managed to update the memory on the e-Paper display one block at a time. Also, we did found a bug[6] in the demo code from the manufacture and that takes us a lot of time until we found out that it is not our mistake that causes the problem.

6 Experiments

6.1 Metrics of Success

1. Basic functions:
   (a) Correct and timely information shown on E-paper sign.
   (b) Able to identify the event creator's tag and unlock the door.
   (c) Support relatively long-distance data transmission between the device and the Raspberry Pi host.
2. Low power usage of the mounted device (E-paper and lock control). We will be using the first fully functional Arduino version as the baseline, and achieve at least 75% less power consumption on our final version.
3. Low latency (ideally smaller than 100ms) between the host and the device.

6.2 Experiment Setup

Based on the above metrics, we decide to perform the following experiments in the final testing. Currently, we have already tested and succeeded in the communication pipe between host and device.

1. Identification experiment: Read wrong keys on RFID readers to see if the door/servo remain locked.
2. Real-time updating experiment: Update calendar to see if the E-paper also change accordingly.
3. Power experiment: Measure the current, voltage, and time of the device under different statuses (starting up, updating display, sending signal, unlocking, sleeping, etc.) together with reasonable assumptions (update frequency, door unlocking frequency, etc.) to estimate the power consumption of the whole system.
4. Latency experiment: Add codes on Raspberry Pi or Arduino to measure the time between sending requests and receiving the later responses.

[6] `https://github.com/waveshare/e-Paper/pull/62`

6.3 Data from Datasheets

Before we actually started the experiments, we looked up for some data from the datasheets to make sure that the values we got from the experiments make sense. Table 3 lists some characteristics of different components.

Parameter	Min	Typ	Max	Unit
Atmega328p-pu				
Power supply current active 4 MHz Vcc = 3 V	-	1.7	2.5	mA
Power supply current active 8 MHz Vcc = 5 V	-	5.2	9	mA
Power-down mode WDT enabled, Vcc = 3 V	-	4.2	8	µA
4.2inch e-Paper Module (B)				
Image update current	-	8	10	mA
Sleep mode current	-	35	50	µA
Image update time at 25 °C	-	12	15	s
nRF24L01+ Wireless Module				
Supply current in power down	-	900	-	nA
(TX) Supply current @ 0 dBm output power	-	11.3	-	mA
(RX) Supply current 1Mbps	-	13.1	-	mA
MFRC522 RFID Reader				
sleep current	-	-	80	µA
operating current	13	26	30	mA
SG90 Servo Motor (4.8V)				
operating speed	0.08	0.09	0.1	sec / 60 °
running current	370	400	430	mA
idle current	5	6	7	mA
LD1117V33 Voltage Regulator				
Quiscent current	-	5	10	mA

Table 3: Characteristics

6.4 Results

6.4.1 Identification Experiment

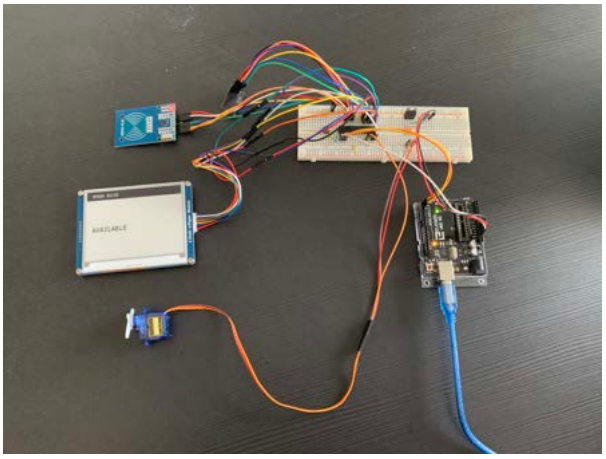

Figure 5: State when the room is available to all.

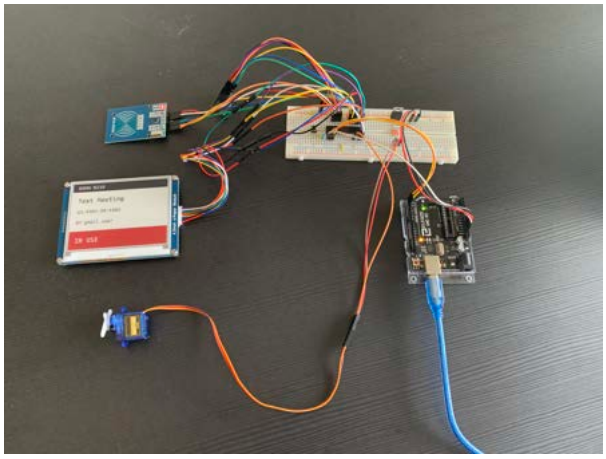

Figure 6: State when the room is being used.

The e-Paper sign displays an available state when no people pre-books the room in default, which is shown in Figure 5. When someone updates the calendar, the e-Paper sign will also update the image to display the in-use state during the reserved time, which is shown in Figure 6. After that, we apply different keys onto the RFID reader and it turn out that only the booking people's key can be authorized to unlock the door. Thus, the identification function is verified for our system.

6.4.2 Real-time Updating Experiment

After we change the schedule on Google calendar, the e-Paper screen can react to show a new correct image in about 10 seconds (the module requires at least 12 seconds for image updating). The circumstance is just similar as the change from the default available state to the in-use state shown in Figure 5 and 6. Hence, the real-timing updating function is verified for our system.

6.4.3 Power Experiment

To better illustrate our power improvement, we measure the current values in idle (delayed / sleep) on the Arduino version, shown in Figure 7, and and on the Atmega328-based version, shown in Figure 8, where a 5V power is supplied in both cases. Read from the multimeter, the sleeping current is 92.9 mA in Arduino case, while it's 9.3 mA in our improved system using Atmega328, which is a **89.99**% optimization in current, same as power since $P = UI$. The power has decreased from 464.4 mW (Arduino) to 46.5 mW (Atmega328) after saving the unneeded circuit part on Arduino board and make use of the sleep / standby modes of the peripheral devices.

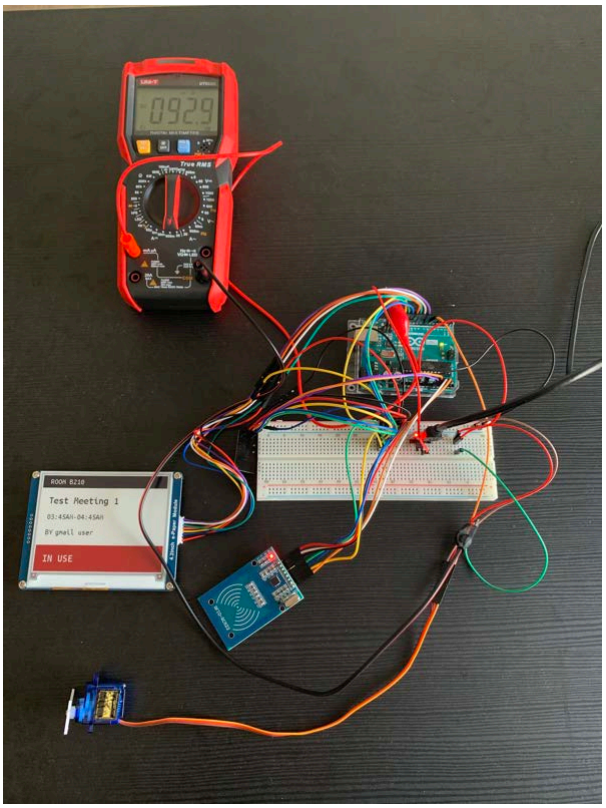

Figure 7: Idle current of the Arduino version

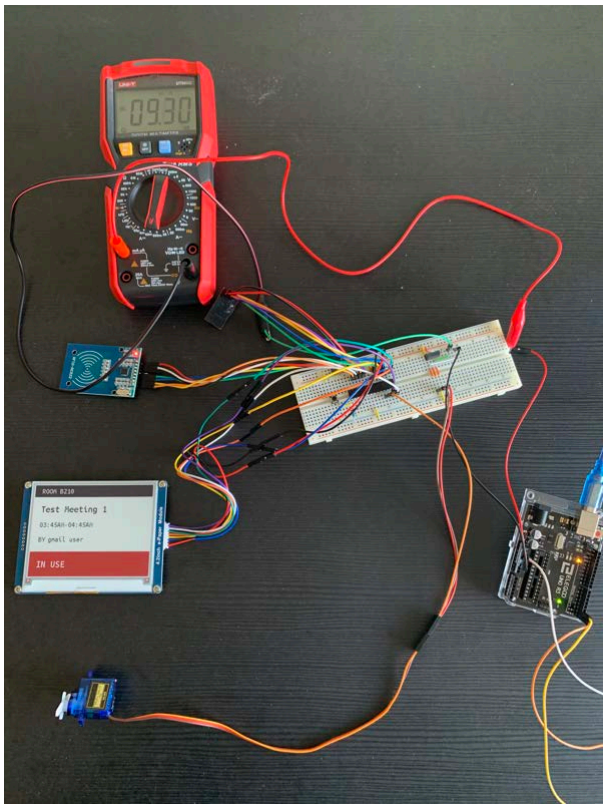

Figure 8: Idle current of the final 328-based version

As for the active power of instantaneous events like RFID check and wireless pulling, we measure the current value by continuous loop execution to help maintain and stablize the instant current change. During RFID check, the

active current is measured as 32 mA and the power is 160 mW after we run it for 1000 times which costs 30466 ms. During wireless pulling, the active current is measured as 26 mA and the power is 130 mW after we run it for 30 times which costs 14610 ms.

In short, the Atmega328 system's power is 46.5 mW during sleeping mode and up to 160 mW when events are activated, improved quite a lot compared with the original design using Arduino.

6.4.4 Latency Experiment

Due to the constraint that the e-Paper module needs 12 to 15 seconds for every updating, the latency measurement recedes into the background here since the value of the latency between host and device is a totally different and much smaller order of magnitude. Therefore, this experiment is skipped.

6.4.5 Wireless Connection Experiment

This extra experiment is performed to check the stability and working range of the wireless connection mainly based on the nRF24L01+ wireless module. We respectively set the distance between device and host as 1m, 2m, 5m and 10m, and both the host and device side can send or receive signals well. Hence, the longest distance between device and host supported by our system should be at least 10m.

6.5 Discussion

From the results, it can be confirmed that we have realized the best solution of this room sign & lock control system based on the hardware/equipment we have. In the experiments that check basic functions, we succeed in verifying the user identification and real-time e-Paper updating. In the power experiments, we compare the power performance for the design using Arduino and the one directly using Atmega328 chip to prove the power efficiency of the final design. We also check the latency and wireless connection: the former is restricted by the e-Paper updating speed and the later supports at least 10 meters in the distance between host and device.

7 Conclusion

7.1 Accomplishment

Finally, we succeed in developing the *Wireless E-paper Room Sign & Door Locking System* as a low-power, low-cost and intelligent system as expected. From the project, we have learned and practised much more skills and thinking about embedded system design beyond the course materials. This comprehensive project includes many arduous challenges from the high-level web servers, where we take advantage of Google APIs to build our real-time system, to the low-level new sensors and actuators, such as the RFID senors and the e-Paper module. As what we have described, great effort in this project is emphasized in the wireless communication and the power optimization: for the former, radio frequency (RF) and Internet is used in user identification (RF) and data transmission (RF and Internet), and for the later, the device circuit connection has been redesigned.

Currently when we look into the development of embedded system, we think there are two irreversible trends on going. First, network usage and concepts about Internet of Things (IoT) will be or is being a significant part in embedded systems. Just like the appliance of Google calendar in this project, future IoT embedded designs will mostly combine with Internet and help establish Cyber-Physical Systems (CPS) in future world. Secondly, based on our experience of replacing the whole Arduino board with its single Atmega328 chip, we think the manufacture

of hardware or chips like CPU will be customized for every unique embedded system due to running performance, power or other factors.

7.2 Future Works

The system still has potential for better improvement in the future, and thus here we raise 3 ideas here for reference.

1. The updating rate can be increased if better e-Paper modules with higher speed are provided.
2. The system can be extended as a one-host-multi-device system, which means one Rpi station can communicate with several mounted devices at the same time.
3. The mechanical part mainly referring to the servo can be replaced more specific and low-power design. Currently, the SG90 Servo motor consumes about 6 mA, while the total current of the system is about 9 mA.