

ELK346E Term Project Report

Dining Hall Registration System Using RFID sensor

İrfan Berdan Öztürk
Buse Soydaş
Sanan Garayev
Egemen Semerci



May 24, 2023

Contents

Abstract	2
1 Introduction	2
2 Materials and Method	2
2.1 Hardware	2
2.1.1 RC522 RFID Sensor	3
2.1.2 Buzzer	4
2.1.3 Servo Motor	4
2.1.4 Potentiometer	4
2.1.5 LED	4
2.2 Software	5
2.2.1 Input	7
2.2.2 Processing	7
2.2.3 Servo Activation	8
2.2.4 Buzzer Alarm	8
3 Implementation and Results	9
4 Discussion and Conclusion	13

Abstract

Currently, as embedded systems are widely utilized for collective places in the universities and schools, the students need much more convenient methods. Traditionally, to take a meal from the hall, the student came, took the meal, and paid with a card or cash. This created long rows, especially in the large-capacity universities like ITU. We created a technological sample of the aforementioned system which takes the ID number from the student card and gives output showing the card ID is true or not and if the system is buggy, manual activation is performed.

1 Introduction

The Dining Hall Registration System project gives the registration system to identify a user by reading user ID from the student card and allow him/her to pass to the dining hall. The system is very similar to one that is being utilized in the dining hall in Istanbul Technical University. The system uses an RFID sensor to read the student cards (tags in our case). If the read ID number is the same as one of pre-registered IDs (predefined in our code), the servo motor is activated which simulates the movement of bars in the real life. Otherwise, the buzzer is activated which is like an alarm system which says the card ID is wrong. Furthermore, if the card is not read by the sensor, the responsible person can activate the second mode with which he/she can manually turn the servo motor via potentiometers.

As hardware, we used STM32L476RG Nucleo mode [1], which is one of well-known nucleo boards in the industry, and as a software tool, we used STM32CubeIDE [2]. Detailed specifications about hardware and software are analyzed in the Materials and Methods section.

2 Materials and Method

2.1 Hardware

For the project, the electronic materials below are used:

1. STM32 L476RG Nucleo Board
2. RC522 RFID Sensor
3. Buzzer 3.3V
4. Mini Servo Motor SG90
5. Potentiometer – 2
6. LED
7. Jumper cables

The system is designed with an RFID sensor, microcontroller chip, servo motor, potentiometer, LED, and buzzer to manage the access of registration. Table 1 shows the pin configuration of the STM32 board and sensors.

STM32 pins	Cube IDE Pins	Type of Element	Element Pins
D14	PB9	RFID	SDA
D13	PA5	RFID	SCK
D11	PA7	RFID	MISO
D12	PA6	RFID	MOSI
NRST		RFID	RST
GND		RFID	GND
3.3 V		RFID	VCC
D7	PA8	Servo	PWM
5 V		Servo	VCC
GND		Servo	GND
A0	PA0	Buzzer	VCC
GND		Buzzer	GND
A1	PA1	Potentiometer 1	PIN
3.3 V		Potentiometer 1	GND
GND		Potentiometer 1	GND
Reset Button	PC13	Interrupt	EXTI-13
A2	PA4	Potentiometer 2	PIN
A5	PC0	LED	VCC

Table 1: Pin Configuration of MCU and Sensors

From Table 1, it can be clearly seen which pins at MCU are connected to which ones in the sensors. During the application of the project, it was noticed that the pin configuration on STM32-L476RG Nucleo board is not same with the pins in Cube IDE Simulation. Therefore, official representation of both board and simulation pins [3] are shown in the Table 1. The first column demonstrates the pins on Nucleo board, the second column shows the pins in the Cube IDE simulation tool, and other parts show the connection of these pins with other electronic elements used in the project.

To understand hardware better, all used materials are explained individually in the section below.

2.1.1 RC522 RFID Sensor

To learn the working principle of an RFID sensor, we first need to understand Serial Peripheral Interface (SPI) protocol.

Serial Peripheral Interface (SPI) is a series communication protocol used in short distance communication [4]. SPI is a synchronous and full-duplex interface. 4 wire SPI devices consist of a clock, chip select (CS), Main Out Sub node (MOSI), and Main In Sub node (MISO). RC522 sensor can also be seen as a 4-wire SPI device. Regarding master-slave

communication, SPI communication requires one master and multiple slaves in the system. In SPI bus, clock controls and synchronizes the transmission of data between master and slave/slaves. If there are more than one slave in the system, CS is activated to select the target device that data need to be transmitted. When it comes to MOSI and MISO, they are data lines. MOSI transmits data from the master to the slave and MISO transmits data from the slave to the master.

In RC522 RFID module, SCK means SPI Clock while SDA shows serial input. RST pin is connected to B0 in STM32 which is normally used to reset the whole system as a part of BOOT0 button.

2.1.2 Buzzer

Buzzer is a simple electronic device which uses DC signal. If there is voltage in the + pin of buzzer it creates a sound which is used as an alarm in our project. Otherwise, it does not create any sound and this means there is no voltage in the + pin. In our project, buzzer is used to demonstrate the differences between the cases, and to show which action is being done at that moment.

2.1.3 Servo Motor

Mini servo motor is used as a way of simulation of bars in the dining hall. A servo motor has a basic electronic structure which consists of GND, VCC, and PWM signal transmitter pin. Servo motors work with PWM principles. This is to say, for instance, if pulse width is 1 ms, servo motor is at 0 degree. If we increase the pulse width to 1.5 ms, servo motor turns to 90 degrees. Arranging pulse width, we can arrange the turning degrees of a servo motor.

2.1.4 Potentiometer

A potentiometer is an electronic device which uses analog signal to work. With a controllable switch, we can change the level of resistance in the circuit which increases or decreases current or voltage values. In our project, we used potentiometer for analog to digital purposes and with potentiometer, we could control LED and buzzer delays.

2.1.5 LED

LED is a simple device which changes the level of light emitted with a given current level. In the project, LED is used to demonstrate the DMA process. The details will be explained in the software section.

Whole parts of hardware and the connections are visualized in the Figure 1 below:

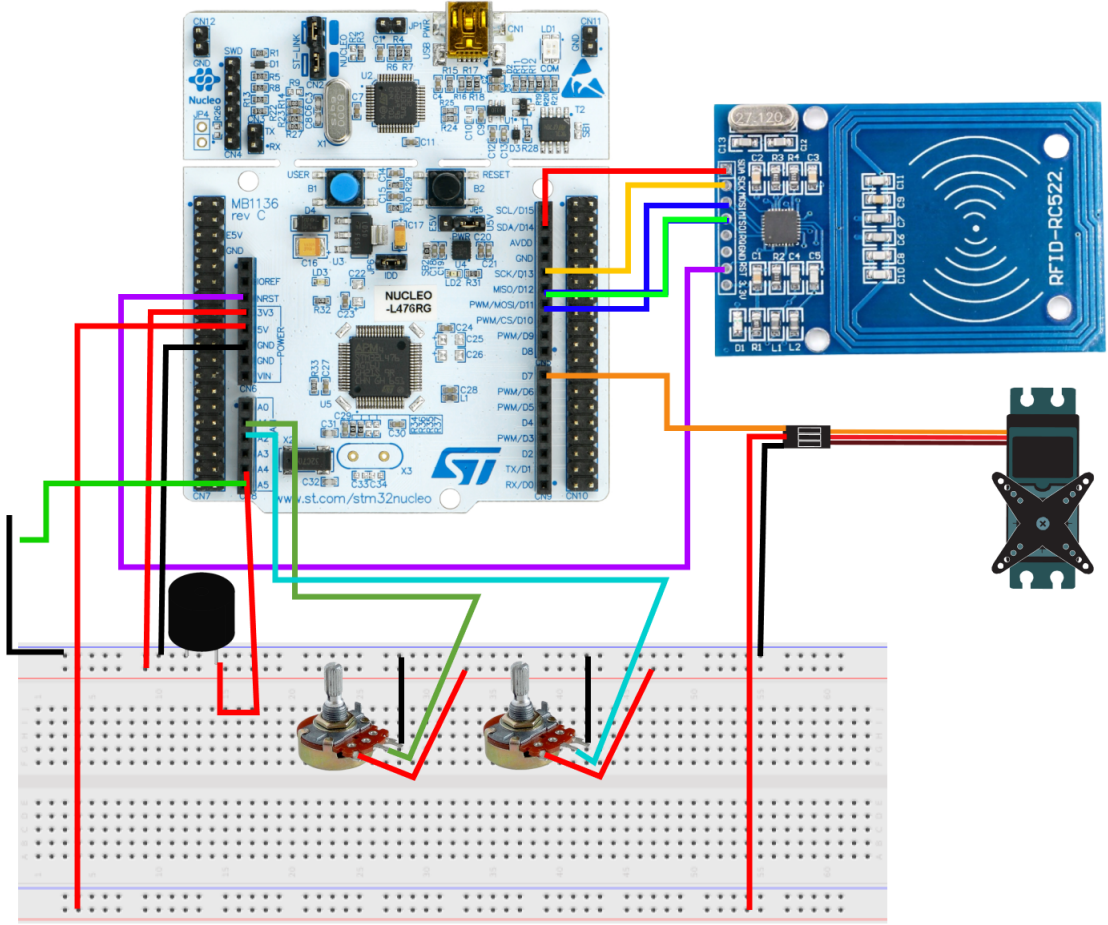


Figure 1: The visualization of connection schema of the circuit in the project

2.2 Software

As a main software tool, STM32CubeIDE is used which utilizes Cube MX in the simulation and configuration part, Eclipse as a main IDE, and Cube Programmer to program the STM card using ST-Link. In the Cube IDE part, all the pin configurations are made, and specific clock values are determined.

Firstly, as we use ST-LINK as main debugger in the SYS part, we chose Serial Debugger. As RFID uses SPI communication protocol, Full-duplex master SPI module is used determining the prescaler to 8 from the Connectivity section. To perform operations, the

internal clock is activated from RCC section as Crystal Ceramic Resonator for High Speed Clock (HSE). For External Interrupt operation we activated EXTI13 pin from the NVIC part.

To control PWM signal, we used TIM2 with PWM Channel 1 from timers section. As clock operation is crucial in PWM, we configured clock signal to give 40 MHz which then delimited to 50 Hz with Prescaler of 800-1 and ARR of 1000.

When it comes to codes, all parts can be seen from the flow diagram from Figure 2.

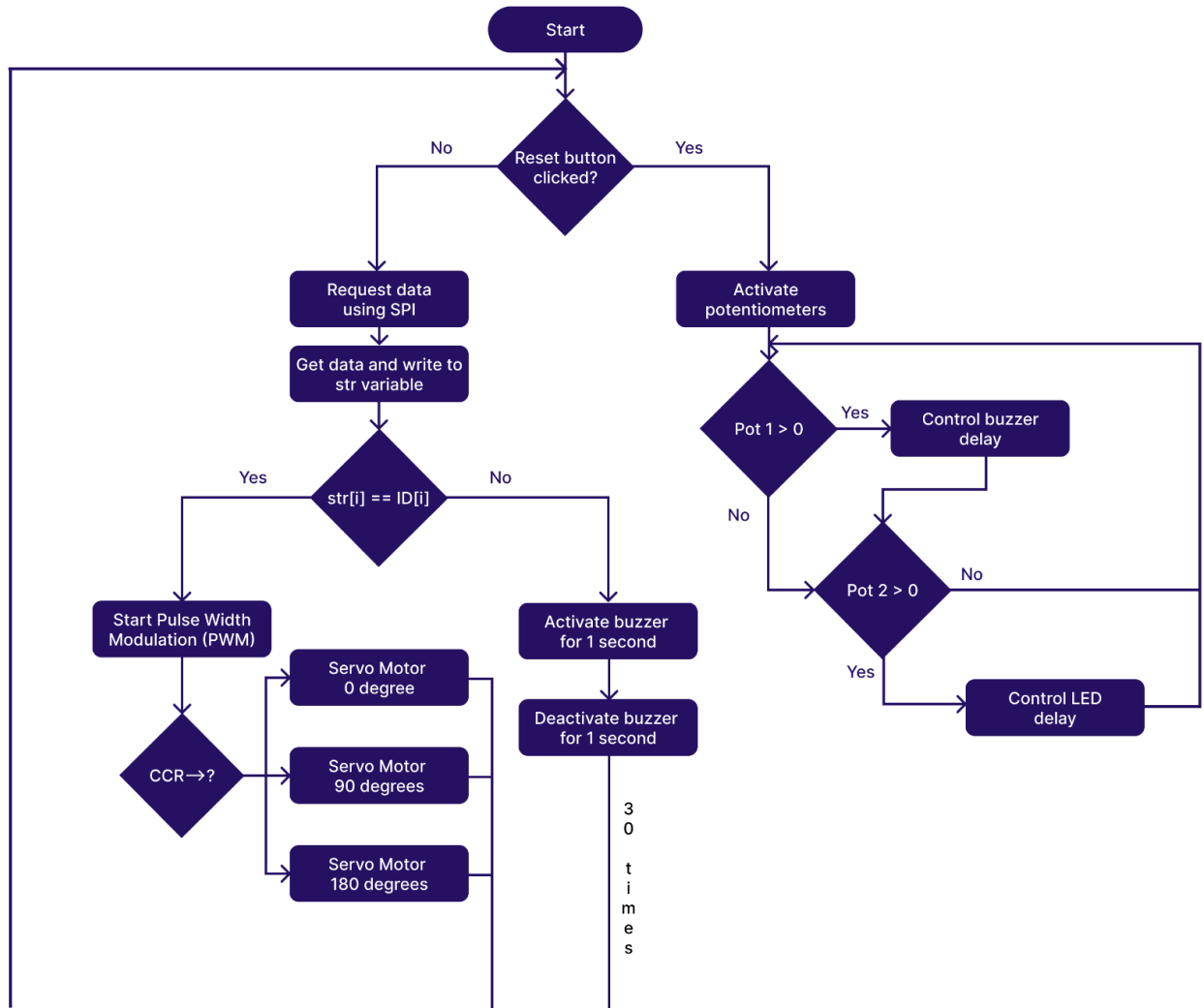


Figure 2: Flow Diagram of the System

The system primarily works based on interrupt command. If the system has started and reset button on the nucleo board is not pushed, the first part is started. At the first part, RFID is activated so that the student can put the card on it and activate the servo motor to go through. This part consists of 4 main sections: Input, Processing, Servo Activation, and Buzzer Alarm.

2.2.1 Input

In this part of the project, RFID sensor is used to collect the user data from the student cards. RFID uses SPI communication with our main MCU (STM32 L476RG) to communicate. The SPI communication is arranged with the algorithms in HAL Library of our STM32 model (L4 Nucleo board). The codes for this part can be seen below:

```
while (1)
{
    /* USER CODE END WHILE */
    // Mode1 activates RFID tag and reading RFID if ID is that of ours servo motor is activates, otherwise the buzzer starts to work like an alarm //
    if (mode == 0)
    {
        status = MFRC522_Request(PICC_REQIDL, str); //MFRC522_Request(0x26, str)
        status = MFRC522_Anticoll(str); //Take a collision, look up 5 bytes
        memcpy(servNum, str, 5); //function for c language:(para1:that place save data,para2:the the source of data,para3:size)
```

As it is seen from the codes, first, we read data via Request function. In the library behind the code, Request function includes the principles of SPI communication. Str is a 5-bit variable, which is used to collect ID of student cards. In each student card or tag, there is a special 5-line ID code which can be expressed in numbers. Str takes those numbers and writes to specified addresses using memcpy function.

2.2.2 Processing

In this stage, collected ID data is processed. As it is seen from the code below, we can check ID with our pre-defined ID number:

```
// In this section if card number is that of ours, the servo motor is activated, otherwise buzzer alarm is activated //
// str keeps the numerical details of ID card
// Numbers like 38, 32, 108, 166, 13 is that of my ID
if((str[0]==38) || (str[1]==32) || (str[2]==108) || (str[3]==166) || (str[3]==13) )
{
    htim1.Instance->CCR1 = 25; // duty cycle is .5 ms
    HAL_Delay(2000);
    htim1.Instance->CCR1 = 75; // duty cycle is 1.5 ms
    HAL_Delay(2000);
    htim1.Instance->CCR1 = 125; // duty cycle is 2.5 ms
    HAL_Delay(2000);
    HAL_Delay(1000);
}
else
{
    int i;
    for (i = 1; i < 30; ++i)
    {
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 0);
        HAL_Delay(1000);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 1);
        HAL_Delay(1000);
    }
}
```


In this code, we process data and notice that if ID is same with the pre-defined ID number, buzzer can be activated with creating sound with 1s delay. Further, this code will be replaced with servo operation.

2.2.3 Servo Activation

If ID is same with the pre-defined ID, we make our servo motor turn by 90 degrees. To arrange that using PWM configuration from HAL Library, we define ms values which are defined in the Hardware section of the report. The codes for servo motor can be seen below:

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
/* USER CODE END 2 */
```

In the section above, PWM channel is activated with the code line of HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1). After that, motor values are noticed.

```
// In this section if card number is that of ours, the servo motor is activated, otherwise buzzer alarm is activated //
// str keeps the numerical details of ID card
// Numbers like 38, 32, 108, 166, 13 is that of my ID
if((str[0]==38) || (str[1]==32) || (str[2]==108) || (str[3]==166) || (str[3]==13) )
{
    htim1.Instance->CCR1 = 25; // duty cycle is .5 ms
    HAL_Delay(2000);
    htim1.Instance->CCR1 = 75; // duty cycle is 1.5 ms
    HAL_Delay(2000);
    htim1.Instance->CCR1 = 125; // duty cycle is 2.5 ms
    HAL_Delay(2000);
    HAL_Delay(1000);
}
```

In the loop above, taking the fact that the cycle time is 20ms in our project, CCR = 25 means 2.5% of duty cycle. After calculation, we can notice that 2.5% of 20 ms is 0.5 ms which corresponds to 0 degrees for servo motor. Calculation CCR of 75 meaning 7.54% of 20 ms which is equal to 1.5 ms, we can rotate the servo motor by 90 degrees in clockwise direction. Finally, CCR of 125 meaning 12.5% of duty cycle makes servo turn by 180 degrees.

2.2.4 Buzzer Alarm

If ID is not same with the one pre-defined, buzzer is activated. Buzzer will be HIGH for 1 second and will be LOW for 1 second to make an alarm effect. According to the code, this alarm will repeat 30 times.

```

else
{
    int i;
    for (i = 1; i < 30; ++i)
    {
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 0);
        HAL_Delay(1000);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 1);
        HAL_Delay(1000);
    }
}

```

Apart from that, when reset button is clicked one time, the External Interrupt starts to work. EXTI is connected to Pin_13 which is reset button on our nucleo board. Clicking once interrupts the code in the first part and starts the second stage. At this stage, potentiometers are activated. The main idea is taken from real life experience like if system is not working, we can manually turn the servo motor. In our project, potentiometers are used to control buzzer and LED due to the issues of servo motor working.

Interrupt function is written as:

```

// Activating interrupt function to perform change between 2 modes
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    mode = mode + 1;
    if(mode>1)
    {
        mode = 0;
    }
}

```

Activating mode 2 starts to collect data taken from potentiometers. Potentiometers normally work with analog signal. However, in our system, we converted these signals to digital signals, which makes it much easier to perform in DC circuit. The first potentiometer is connected to a buzzer. Increasing the value of potentiometer increases the delay time between buzzer activation and deactivation.

The second potentiometer is connected to LED light which also controls the delay time between switching on and off the LED. There is one question coming through: are these potentiometers work at the same time? The answer is yes. To do that, we use two channels at the same time with the help of Direct Memory Access (DMA) [5] architecture. In our system, ADC is performed using DMA which uses two inputs from ADC1.

3 Implementation and Results

The results of the project are shown step by step in the section below.

Step 1: Hardware Setup

Predefined hardware setup is implemented on real board using NUCLEO-L476RG model and other electronic components which can be seen from Figure 3.

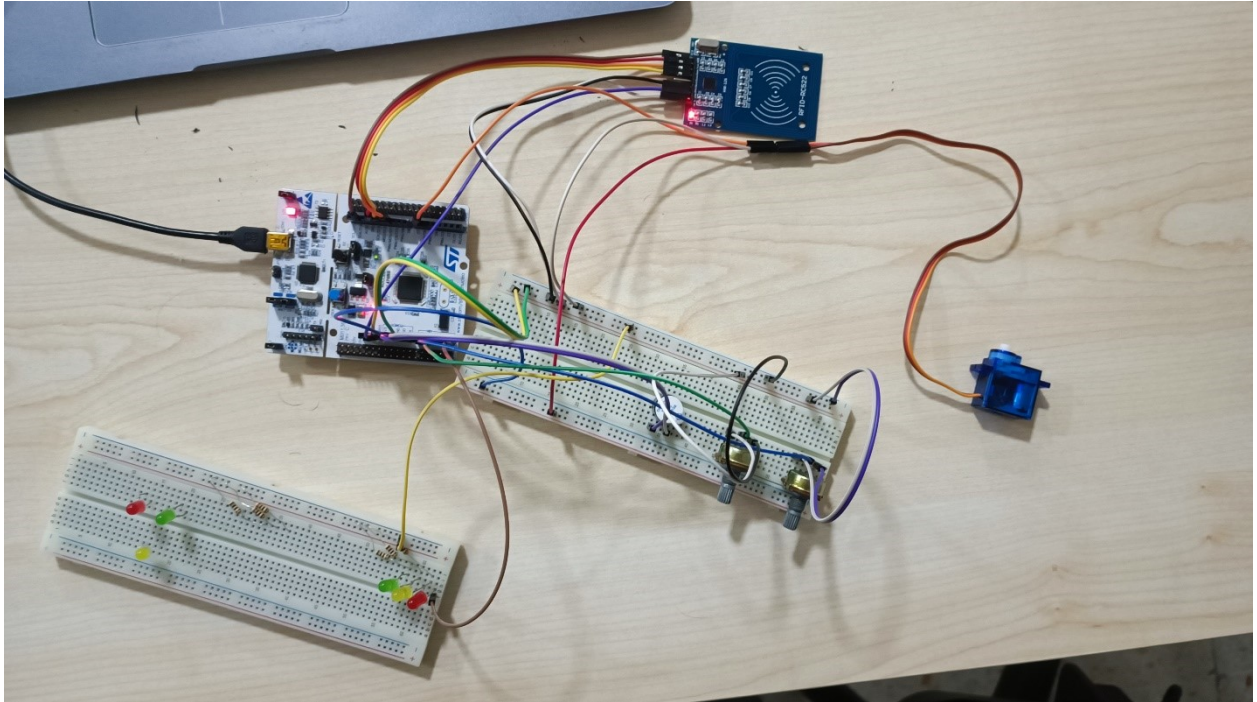


Figure 3: Actual hardware setup

Step 2: CubeMX configuration

In this section CubeMX details are explained. As it is seen from Figure 4, all pins are configured beforehand. Potentiometer pins are joined to ADC channels using DMA, PC13 is connected as external interrupt. Serial wire and SPI configuration are made and clock frequency is changed to 40 MHz. Clock configuration can be seen from Figure 5.

Expression	Type	Value
▼ serNum	unsigned char [5]	[5]
0= serNum[0]	unsigned char	147 '\223'
0= serNum[1]	unsigned char	32 ''
0= serNum[2]	unsigned char	0 '\0'
0= serNum[3]	unsigned char	0 '\0'
0= serNum[4]	unsigned char	0 '\0'
▼ str	unsigned char [16]	[16]
0= str[0]	unsigned char	147 '\223'
0= str[1]	unsigned char	32 ''
0= str[2]	unsigned char	0 '\000'
0= str[3]	unsigned char	0 '\000'
0= str[4]	unsigned char	0 '\000'
0= str[5]	unsigned char	0 '\000'
0= str[6]	unsigned char	0 '\000'
0= str[7]	unsigned char	0 '\000'
0= str[8]	unsigned char	0 '\000'
0= str[9]	unsigned char	0 '\000'
0= str[10]	unsigned char	0 '\000'
0= str[11]	unsigned char	0 '\000'
0= str[12]	unsigned char	0 '\000'
0= str[13]	unsigned char	0 '\000'
0= str[14]	unsigned char	0 '\000'
0= str[15]	unsigned char	0 '\000'

Figure 6: RFID Variables Expression

After reading the variables from the RFID tag, servo motor is activated with degrees of 90 and 180 degrees. The sample representation can be seen from Figure 7.

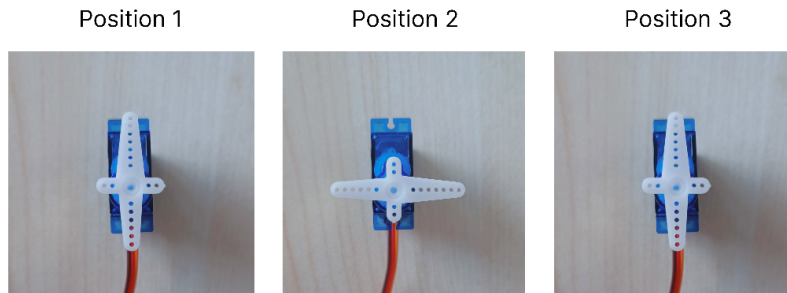


Figure 7: Servo motor representation

Step 4: Potentiometer control of buzzer and LED

In this part mode 1 is activated using external interrupt. First potentiometer controls buzzer delay and the second one controls LED delay. We read analog input from potentiometer and convert the signal to DC to activate buzzer and LED. Using DMA two potentiometers work at the same time. The potentiometer values can be seen in live

expressions section which is represented in Figure 8. Data[0] shows the values of potentiometer 1 and data[1] shows potentiometer 2.




Expression	Type	Value
>  serNum	unsigned char [5]	[5]
>  str	unsigned char [16]	[16]
(*)= data[0]	uint32_t	387
(*)= data[1]	uint32_t	73
 Add new expression		

Figure 8: Potentiometer values

4 Discussion and Conclusion

In this report, the simulation of the real-life dining hall student registration system is performed. RFID sensor is used to take data and servo and buzzer are used to show the correct output to the user.

It can be concluded that for RFID projects, STM32 is working very well. In the project, as it is shown that multitasking capabilities, timer options, memory access and using different communication protocols make STM32 much superior. However, in modern IoT systems, the connection with the Internet becomes crucial. Especially, as we are turning to QR technology rather than using traditional cards, we need to use boards which can connect to the databases on the Internet like Firebase, etc. In this case, STM32 is not enough, and we may need to use more complex circuits like Raspberry Pi or STM like boards such as ESP32.

Furthermore, as we used clone model of STM32, we encountered many different problems such as ST-Link connection, different chip configuration, etc. To overcome these problems, we can use original Nucleo boards or as mentioned, we can change the chip to ESP32 which is much easier to program even with Arduino IDE.

References

- [1] “Stm32 nucleo-64 development board with stm32l476rg mcu, supports arduino and st morpho connectivity.” <https://www.st.com/en/evaluation-tools/nucleo-l476rg.html>.
- [2] “Integrated development environment for stm32.” <https://www.st.com/en/development-tools/stm32cubeide.htm>.
- [3] “Nucleo-l476rg.” <https://os.mbed.com/platforms/ST-Nucleo-L476RG/>.
- [4] Person, “Introduction to spi interface.” <https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html>.
- [5] “Getting started with dma.” https://wiki.st.com/stm32mcu/wiki/Getting_started_with_DMA.