AUTHENTICATED VOTING SYSTEM USING RFID

EE6352 EMBEDDED SYSTEM DESIGN

Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna Sri Lanka

Group Members:

EG/2020/3967
EG/2020/3974
EG/2020/3975
EG/2020/3976
EG/2020/3977

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 INTRODUCTION TO THE PROBLEM OR THE SOLUTION

In today's democratic landscape, ensuring the fairness and security of elections is crucial. When considering the current voting systems, they face various problems that can affect their integrity, efficiency and public trust.

One of the main problem is election hacking. Vulnerabilities in voting can be exploited, leading to potential tampering with vote counts. There are some instances reported voting using false identities.

In additionally to that, there are problems with delays, loss of votes and incorrect voting counts. Inadequate transparency in the voting process can lead to distrust in the electoral outcomes. Further prolonged counting processes can cause uncertainty and suspicion among the electorate.

Further in some countries, resource allocation for the election is an economically considerable problem. Inadequate funding and resources can hinder the smooth functioning of elections. Further considering the polling management systems due to the mismanagement, there may be long wait times, confusion, and frustration among voters.

In order to address this issue, our project provides an RFID based on authenticated voting system. Our goal is to develop a voting system that ensures that only qualified voters may cast votes using RFID tags. This method simplifies the voting process while enhancing security.

The main goal of our project is to design a voting machine that prevents unauthorized voting and electoral cheating. Through RFID tags assigned to registered voters, we establish a reliable way to verify their identities. This helps to maintain the integrity of the voting process by ensuring only legalized votes are counted.

Our main goal is to provide a simple yet effective solution to the complexities of modern elections. In an increasingly digital environment, we work to preserve democratic values and guarantee the integrity of the electoral process by utilizing RFID technology.

1.2 EXISTING SOLUTIONS

There are some electronic voting machines designed for the voting purpose in some countries. [1]

1. Optical Scanning
   In an optical scan voting system, also known as marksense voting, voters indicate their selections on one or more paper ballots. These ballots are then fed into a scanner, which generates an electronic image of each ballot. The scanner interprets these images, counts the votes for each candidate, and typically saves the images for future verification.

   Voters can mark their choices directly on the paper in designated areas for each candidate. Alternatively, they can make their selections on an electronic screen, which prints the chosen names along with a bar code or QR code that summarizes all selections onto a paper sheet. This paper sheet is then submitted to the scanner.

2. Direct Recording Electronic(DRE)
   In a Direct Recording Electronic (DRE) voting machine system, voters use a touch screen to select their choices, which they can change as many times as they want before finalizing their vote. Election staff initialize the machine for each voter to prevent multiple votes. The voting data is stored in memory components and can be extracted at the end of the election.

   Some DRE machines also print the names of the selected candidates on paper for the voter to verify, although less than 40% of voters do so. These printed names are displayed behind glass in the

machine and can be used for audits and recounts if necessary. The voting data tally is printed at the end of the paper tape, known as the Voter-verified paper audit trail (VVPAT). Tallying VVPATs requires approximately 20-43 seconds of staff time per vote.

1.3 DRAWBACKS OF EXISTING SOLUTIONS

Problems with Optical Scanning: [1]

Numerous errors have been discovered in optical scan systems. These include issues such as ballots being fed upside down, multiple ballots being pulled through simultaneously during central counts, paper jams, and malfunctioning sensors that can become blocked or overheated, leading to incorrect interpretation of ballots. Additionally, printing misalignments with the programming, programming errors, and file losses have been reported. The exact causes of these programming errors are seldom identified, making it unclear whether they were accidental or intentional.

Problems with Direct Electronic Recording: [1]

For machines lacking a Voter-Verified Paper Audit Trail (VVPAT), there is no way to verify individual votes. Machines with VVPATs present additional challenges: verifying votes is more costly than with traditional paper ballots. This is because the flimsy thermal paper, printed in long continuous rolls, can cause staff to lose their place easily, and the printouts include every change made by voters, not just their final choices.

Issues have arisen such as public web access to the software before it is loaded into the machines for elections, and programming errors that result in votes being assigned to incorrect candidates. The Federal Constitutional Court of Germany ruled that existing machines could not be allowed as they could not be publicly monitored.

Moreover, successful hacks of these machines have been demonstrated under laboratory conditions.

1.4 OBJECTIVE

Authenticated Voting System using RFID aims to achieve the following objectives,

1. System Security and Authentication:
Implement a secure authentication mechanism using RFID technology to ensure that only authorized voters can cast their votes if the authentication is invalid, system should not be allowed to vote. Once voted, a voter should not be allowed to vote again.

2. User Interface and Interaction:
Design an intuitive user interface that provides clear instructions and displays candidate ID.

3. Button Interaction and Feedback:
Develop a button handling module that accurately detects button presses/releases and triggers appropriate actions, providing feedback through LED indicator.

4. System Reliability and Performance:
Create a reliable system that operates efficiently, with fast RFID tag detection and authentication, minimal delays in the voting process, and robust error handling.

5. Data Integrity and Privacy:
Maintain data integrity by securely storing voter information on RFID tags, protecting against unauthorized access, and ensuring compliance with data privacy regulations.

6. Scalability and Cost-Efficiency:

Design a scalable system architecture that can accommodate many voters while considering the cost-effectiveness of RFID readers, tags, backend systems, and maintenance.

1.5 PROJECT SCOPE

There are some limitations of this system when considering the practical usage with it.

The main concern is about RFID tag management. There may be issues related to the distribution and issuance of RFID tags. Managing the distribution and issuance of RFID tags for the eligible voters may be complex when considering about large scaled elections. There it is needed to ensure that each voter receives the correct tag and otherwise there may be problems with damaged or misplaced RFID tags. If there are duplicated tags, then it cannot be detected by the system as if the microcontroller is not programmed to detect the tag number.

There may be problem regarding the readability and interference. The reliability of the RFID tag detection depends on the sensitivity and quality of the RFID reader. If a low quality RFID reader is used, it may face to struggles when detecting the tags consistently, that may cause to authentication failures and delays in voting process. Further interferences from other electronic and related devices or some environmental factors can disturb to the RFID communication, and finally it can affect to the readability of tags.

There are some security concerns about the project. For example, storing voter information in RFID tags may affect to the data privacy and protection and further they can be used for unauthorized access to the voter data.

There are some considerations about the scalability and cost. When it comes to the real system's infrastructure, a considerable cost should be paid to RFID readers, tags and for data management and processing using backend systems. Further for regular maintenance and upkeep should be there to ensure the system's reliability and security. And further when scaling up the system the components which are using should be replaced with some advanced components which costs than the simple implementation.

Lastly, there are some legal and regulatory requirements that should be considered when using the system in advanced elections. For that, the necessary requirements also should be combined with the normal system in order to compliance with regulations.

2 SPECIFICATIONS

The specifications of the project can be mentioned as follows:

1. **RFID Authentication**: The system uses RFID technology to authenticate voters. Only authorized voters, identified through their unique RFID tags, can cast their votes. This ensures that the voting process is secure and tamper-proof, reducing the risk of fraudulent voting practices

2. **Display**: The system features a user interface with a 16x2 LCD display. This display shows voter ID, instructions to follow and the results at the end of the election. Additionally, LED indicators provide visual feedback during the voting process, enhancing user experience and interaction.

3. **Button-Based Voting**: Voters cast their votes by pressing corresponding buttons assigned to each candidate. The system accurately records the votes in real-time on the LCD screen, ensuring transparency and immediate feedback for voters.

4. **Automatic Vote Counting**: The microcontroller automatically tallies the votes and updates the count for each candidate. This eliminates the need for manual counting, reducing human errors and increasing the accuracy and efficiency of the voting process.

5. **Scalability**: The system is designed to be scalable, accommodating a varying number of candidates and voters. This is achieved by adjusting program parameters and hardware configurations, making the system suitable for both small-scale and large-scale elections.

6. **Data Integrity and Privacy**: The system securely stores voter information on RFID tags, protecting against unauthorized access and ensuring compliance with data privacy regulations. This maintains the integrity of the voting data and protects voter privacy.

7. **System Reliability**: The system is built to operate efficiently, with fast RFID tag detection and authentication, minimal delays in the voting process, and robust error handling. This ensures a smooth and reliable voting experience. When invalid RFID tag is read with the module, it should not allow to vote and authenticated RFID tag should be allowed to use only once with system to avoid repeat voting. When one voting process is ongoing, another person should not be allowed to access the system by using his/her RFID tag.

8. **Component Integration**: The system integrates various components, including an RFID reader module, LCD display, LED indicators, push buttons, capacitors, and an Arduino Uno (ATmega328P) microcontroller. These components work together seamlessly to provide a functional and reliable voting system.

9. **Results Interpretation:** At the end of the election, the results of the election is displayed only when the owner's RFID tag is entered to the system.

3 IMPLEMENTED SOLUTION

3.1 BLOCK DIAGRAM OF THE FINAL PRODUCT

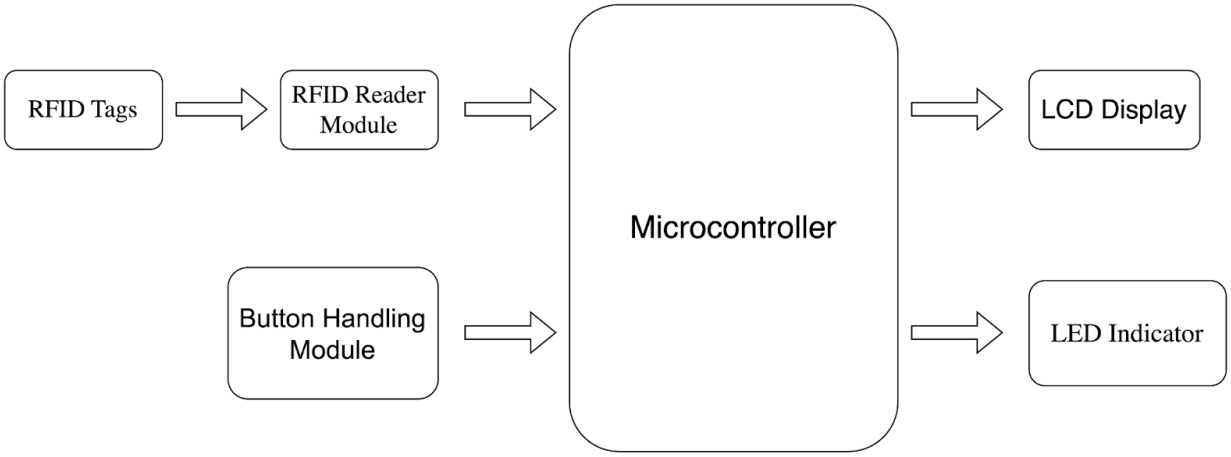

Figure 3-1 Block Diagram of the Final Product

3.1.1 DETAILED DESCRIPTION

The function or the detailed description of each block of the block diagram which is under the Figure 3-1 can be described as follows:

1. Microcontroller – Atmega328P

ATmega328P microcontroller serves as the central processing unit controlling and coordinating all components. It controls all the other blocks and peripherals in the system. The microcontroller handles tasks such as reading button inputs, receiving data from the RFID reader module, processing this data, and updating the LCD display accordingly.

2. RFID Reader Module – EM18

When an RFID tag is scanned by the RFID reader module, it receives data containing information about the RFID tags scanned by the reader. It validates the received RFID data to determine if it corresponds to an authorized RFID tag. Upon successful validation, it signals the microcontroller to allow the user to cast a vote.

3. LCD Display

The LCD display module receives data from the microcontroller and updates the LCD display accordingly. It displays information such as candidate names, their respective vote counts, system messages, and instructions for the user. Additionally, it handles tasks such as initializing the display, sending commands and data to it, and formatting text for display.

4. Button Handling Module

This block interfaces with the buttons connected to the microcontroller. It detects button presses and releases and processes them accordingly. Upon detecting a button press, it triggers the corresponding action, such as displaying the total vote count for a candidate or resetting all vote counts.

3.2 MODEL SOLUTION

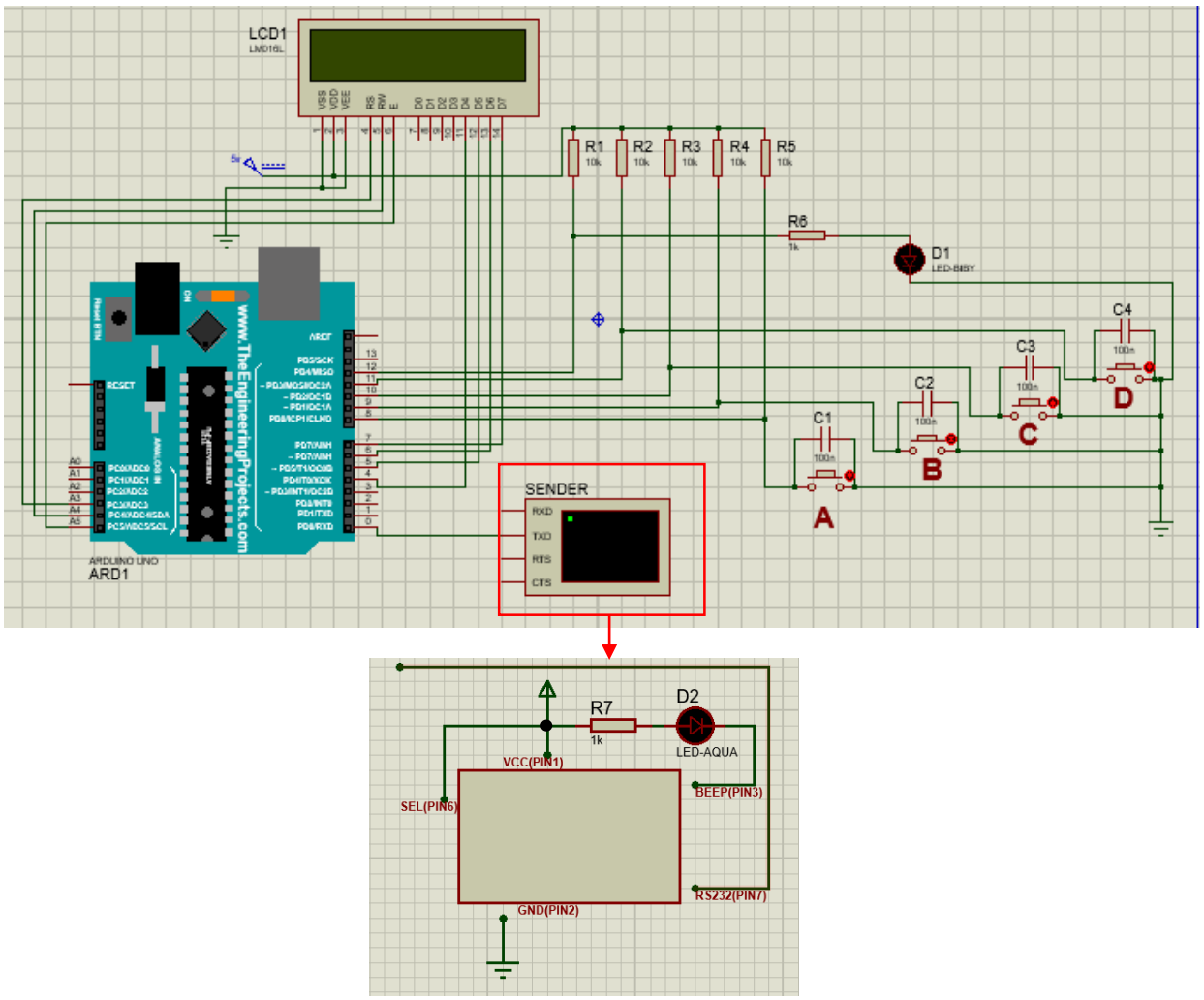

Figure 3-2 Proteus Simulation Diagram

The above Figure 3-2 shows the Proteus simulation diagram for the circuit. When simulating the circuit, a virtual terminal has been used instead of EM18 reader which is same as connected to the RX port. Instead of typing the input in proteus simulation, in real simulation the RFID tag can be passed just interfacing the card in front of the reader module.

3.2.1 CODE

main.c

```
/*
 * EmbeddedSystemsProject.c
 *
 * Created: 5/6/2024 5:36:27 AM
 * Author : IYENSHI
 */

#define F_CPU 16000000UL

// Define baud rate
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include "usart.h"
#include "lcd.h"
```

```
#ifndef bit_is_clear
#define bit_is_clear(sfr, bit) (!(sfr & (1 << bit)))
#endif


int main() {

    DDRB = 0b11110000;          // Initialize Port B pin 0,1,2,3 as inputs
    //DDRB &= 0b11110111;               // Initialize Port B pin 4 as output

    int validation = 0;         //     This is used to validate the RFID tag
    int voted = 0;                      //     This is used to check whether that vote has been
given

    // This is used to calculate votes and display it

    int16_t voteA = 0;
    int16_t voteB = 0;
    int16_t voteC = 0;
    int16_t voteD = 0;

    char A [4];
    char B [4];
    char C [4];
    char D [4];

    //Define valid RFID tags
    const char authenticated_RFID[5][12] =
{{'4','9','0','0','1','2','C','6','9','1','0','C'},{'4','9','0','0','1','4','3','8','4','B','2
','E'},{'4','9','0','0','1','C','C','3','4','C','D','A'},{'4','9','0','0','1','7','0','9','6',
'B','3','C'},{'4','9','0','0','1','2','8','5','B','D','6','3'}};

    //Defined controlled RFID
    const char controlled_RIFD[12] ={'4','9','0','0','1','1','8','6','0','B','D','5'};

    //Initialize individual's voting to prevent giving more than one vote
    int votingCalculation[5] = {0, 0, 0, 0, 0};


    uart_init();  // Initialize UART
    Lcd_Init();          // Initialize LCD
    Lcd_Clear();


    // Main loop
    while (1) {
           Lcd_Set_Cusor(0,0);                      // Locate cursor to start point
           Lcd_Write_String("RFID TAG :"); // Display
           Lcd_Set_Cusor(1,0);                      // Locate cursor in next line

           char inserted_RFID[12];     // Define an array to store user voting

           enableRX();

           for (int i=0;i<12;i++){
                  inserted_RFID[i] = uart_receive(); // Receive character from UART
           }

           disableRX();

           Lcd_Write_String(inserted_RFID); // Display RFID tag inserted
           _delay_ms(3000);                          // For 3 s

           Lcd_Clear();                              // Clear display

           // Check the input tag is matched with authenticated tag
           for (int j=0;j<5;j++)
```

```
		{
			// When found
			if
((inserted_RFID[0]==authenticated_RFID[j][0])&(inserted_RFID[1]==authenticated_RFID[j][1])&(in
serted_RFID[2]==authenticated_RFID[j][2])&(inserted_RFID[3]==authenticated_RFID[j][3])&(insert
ed_RFID[4]==authenticated_RFID[j][4])&(inserted_RFID[5]==authenticated_RFID[j][5])&(inserted_R
FID[6]==authenticated_RFID[j][6])&(inserted_RFID[7]==authenticated_RFID[j][7])&(inserted_RFID[
8]==authenticated_RFID[j][8])&(inserted_RFID[9]==authenticated_RFID[j][9])&(inserted_RFID[10]=
=authenticated_RFID[j][10])&(inserted_RFID[11]==authenticated_RFID[j][11]))
			{
				if(votingCalculation[j] == 0){ // Checked that tag has already voted,
if not then,
					PORTB|=(1<<PINB4);		// Turn on LED at pin 4
					validation = 1;				// Set tag as valid;
					votingCalculation[j] = 1;   // Set tag as voted
				} else {
					validation = 2;				// This is used to
identify that tag has already voted
				}
				break;
			}
		}

		if(validation == 0){
			if
((inserted_RFID[0]==controlled_RIFD[0])&(inserted_RFID[1]==controlled_RIFD[1])&(inserted_RFID[
2]==controlled_RIFD[2])&(inserted_RFID[3]==controlled_RIFD[3])&(inserted_RFID[4]==controlled_R
IFD[4])&(inserted_RFID[5]==controlled_RIFD[5])&(inserted_RFID[6]==controlled_RIFD[6])&(inserte
d_RFID[7]==controlled_RIFD[7])&(inserted_RFID[8]==controlled_RIFD[8])&(inserted_RFID[9]==contr
olled_RIFD[9])&(inserted_RFID[10]==controlled_RIFD[10])&(inserted_RFID[11]==controlled_RIFD[11
]))
			{
				Lcd_Write_String("VOTING ENDED");
				Lcd_Set_Cusor(1, 0);
				Lcd_Write_String("RESULTS =>");
				_delay_ms(3000);
				Lcd_Clear();
				Lcd_Set_Cusor(0, 0);

				Lcd_Write_String ("A=");
				itoa(voteA,A,10);
				Lcd_Write_String(A);
				Lcd_Set_Cusor(0, 8);

				Lcd_Write_String ("B=");
				itoa(voteB,B,10);
				Lcd_Write_String(B);
				Lcd_Set_Cusor(1, 0);

				Lcd_Write_String ("C=");
				itoa(voteC,C,10);
				Lcd_Write_String(C);
				Lcd_Set_Cusor(1, 8);

				Lcd_Write_String("D=");
				itoa(voteD,D,10);
				Lcd_Write_String(D);

				break;
			}else{
				_delay_ms(1000);
				Lcd_Write_String("INVALID TAG");   // Display
				validation = 0;
				_delay_ms(2000);                                    // for 2 s
				Lcd_Clear();                         // Clear display
```

```
                }
        }

        if (validation == 2)                            // If already voted
        {
                _delay_ms(1000);
                Lcd_Write_String("ALREADY VOTED");  // Display
                validation = 0;
                _delay_ms(2000);                                // for 2 s
                Lcd_Clear();                                    // Clear display
        }



        if (validation == 1)                            // If valid
        {
                _delay_ms(1000);
                Lcd_Write_String("YOU CAN VOTE NOW");
        }

        while (validation == 1)                                 // Until tag is identified
as valid
        {
                while(voted == 0)                               // Until tag is not
identified as voted
                {
                        if (bit_is_clear(PINB,0))       // If button A is pressed
                        {
                                voteA++;                                // Increase vote for
A by 1
                                validation = 0;                 // Set now tag as not
valid
                                voted = 1;                              // Set now tag as
voted
                        }

                        if (bit_is_clear(PINB,1))       // If button B is pressed
                        {
                                voteB++;                                // Increase vote for
B by 1
                                validation = 0;                 // Set now tag as not
valid
                                voted = 1;                              // Set now tag as
voted
                        }

                        if (bit_is_clear(PINB,2))       // If button C is pressed
                        {
                                voteC++;                                // Increase vote for
C by 1
                                validation = 0;                 // Set now tag as not
valid
                                voted = 1;                              // Set now tag as
voted
                        }

                        if (bit_is_clear(PINB,3))       // If button D is pressed
                        {
                                voteD++;                                // Increase vote for
D by 1
                                validation = 0;                 // Set now tag as not
valid
                                voted = 1;                              // Set now tag as
voted
                        }
                }
```

```
                if (validation == 0)                        // If the vote has been given
                {
                        voted = 0;                                          // Now set voted as 0
for previous cases
                        Lcd_Clear();                                // Clear display
                        Lcd_Write_String ("THANK U FOR VOTE"); // Display
                        _delay_ms(2000);                            // For 2 s
                        PORTB&=~(1<<PINB4);                         // Turn off LED

                        Lcd_Clear();                                // Clear display

                }
            }
    }
    return 0;
}
```

lcd.h

```
/*
 * lcd.h
 *
 * Created: 5/9/2024 7:00:00 PM
 * Author: IYENSHI
 */

#ifndef LCD_H_
#define LCD_H_

#include <avr/io.h>

#define DATA_BUS  PORTD
#define CTL_BUS          PORTC
#define DATA_DDR  DDRD
#define CTL_DDR          DDRC
#define LCD_D4                  4
#define LCD_D5                  5
#define LCD_D6                  6
#define LCD_D7                  7
#define LCD_EN                  5
#define    LCD_RW               4
#define    LCD_RS               3
#define LCD_CMD_CLEAR_DISPLAY            0x01
#define LCD_CMD_DISPLAY_NO_CURSOR          0x0c
#define LCD_CMD_DISPLAY_CURSOR_BLINK       0x0F
#define LCD_CMD_4BIT_2ROW_5X7              0x28

void Lcd_Init(void);
void Lcd_Send_Command (uint8_t );
void Lcd_Clear(void);
void Lcd_Set_Cusor(uint8_t , uint8_t );
void Lcd_Write_Str(uint8_t characte);
void Lcd_Write_String(const char *str);

// Initialize LCD
void Lcd_Init(void)
{
    DATA_DDR = (1<<LCD_D7) | (1<<LCD_D6) | (1<<LCD_D5)| (1<<LCD_D4);
    CTL_DDR |= (1<<LCD_EN)|(1<<LCD_RW)|(1<<LCD_RS);
    DATA_BUS = (0<<LCD_D7)|(0<<LCD_D6)|(1<<LCD_D5)|(0<<LCD_D4);
    CTL_BUS|= (1<<LCD_EN)|(0<<LCD_RW)|(0<<LCD_RS);
    _delay_ms(1);
    CTL_BUS &=~(1<<LCD_EN);
    _delay_ms(1);
    Lcd_Send_Command(LCD_CMD_4BIT_2ROW_5X7);
    _delay_ms(1);
```

```
    Lcd_Send_Command(LCD_CMD_DISPLAY_CURSOR_BLINK);
    _delay_ms(1);
    Lcd_Send_Command(0x80);
}

// Send commands
void Lcd_Send_Command (uint8_t command)
{
    DATA_BUS=(command&0b11110000);
    CTL_BUS &=~(1<<LCD_RS);
    CTL_BUS |=(1<<LCD_EN);
    _delay_ms(1);
    CTL_BUS &=~(1<<LCD_EN);
    _delay_ms(1);
    DATA_BUS=((command&0b00001111)<<4);
    CTL_BUS |=(1<<LCD_EN);
    _delay_ms(1);
    CTL_BUS &=~(1<<LCD_EN);
    _delay_ms(1);
}

// Display one character
void Lcd_Write_Str(uint8_t character)
{
    DATA_BUS=(character & 0b11110000);
    CTL_BUS|=(1<<LCD_RS);
    _delay_us(1);
    CTL_BUS |=(1<<LCD_EN);
    _delay_ms(20);
    CTL_BUS &=~(1<<LCD_EN);
    _delay_ms(20);
    DATA_BUS=((character & 0b00001111)<<4);
    _delay_ms(20);
    CTL_BUS |=(1<<LCD_EN);
    _delay_ms(20);
    CTL_BUS &=~(1<<LCD_EN);
}

// Clear display
void Lcd_Clear(void)
{
    Lcd_Send_Command(LCD_CMD_CLEAR_DISPLAY);
}

// Set cursor
void Lcd_Set_Cusor (uint8_t line,uint8_t pos)
{
    Lcd_Send_Command((0x80|(line<<6))+pos);
    _delay_us (50);
}

// Display string
void Lcd_Write_String(const char *str){
    while (*str) {
           Lcd_Write_Str(*str++);
           _delay_us(500);
    }
    Lcd_Send_Command(LCD_CMD_DISPLAY_NO_CURSOR);
}

#endif /* LCD_H_ */
```

usart.h

```
/*
 * usart.h
 *
 * Created: 5/10/2024 5:22:43 AM
```

```
 *  Author: IYENSHI
 */

#ifndef USART_H_
#define USART_H_

#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)

void uart_init();
void uart_transmit(unsigned char data);
unsigned char uart_receive();
void enableRX();
void disableRX();

// Function to initialize UART communication
void uart_init() {
    // Set baud rate
    UBRR0H = (uint8_t)(BAUD_PRESCALE >> 8);
    UBRR0L = (uint8_t)(BAUD_PRESCALE);
    // Set frame format: 8 data bits, 1 stop bit, no parity
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

// Function to receive a character via UART
unsigned char uart_receive() {
    // Wait for data to be received
    while (!(UCSR0A & (1 << RXC0)));
    // Get and return received data from buffer
    return UDR0;
}


void enableRX() {
    UCSR0B |= (1 << RXEN0);
}

// Function to disable RX
void disableRX() {
    UCSR0B &= ~(1 << RXEN0);
}

#endif /* USART_H_ */
```

3.3 FLOW CHART OF THE FIRMWARE

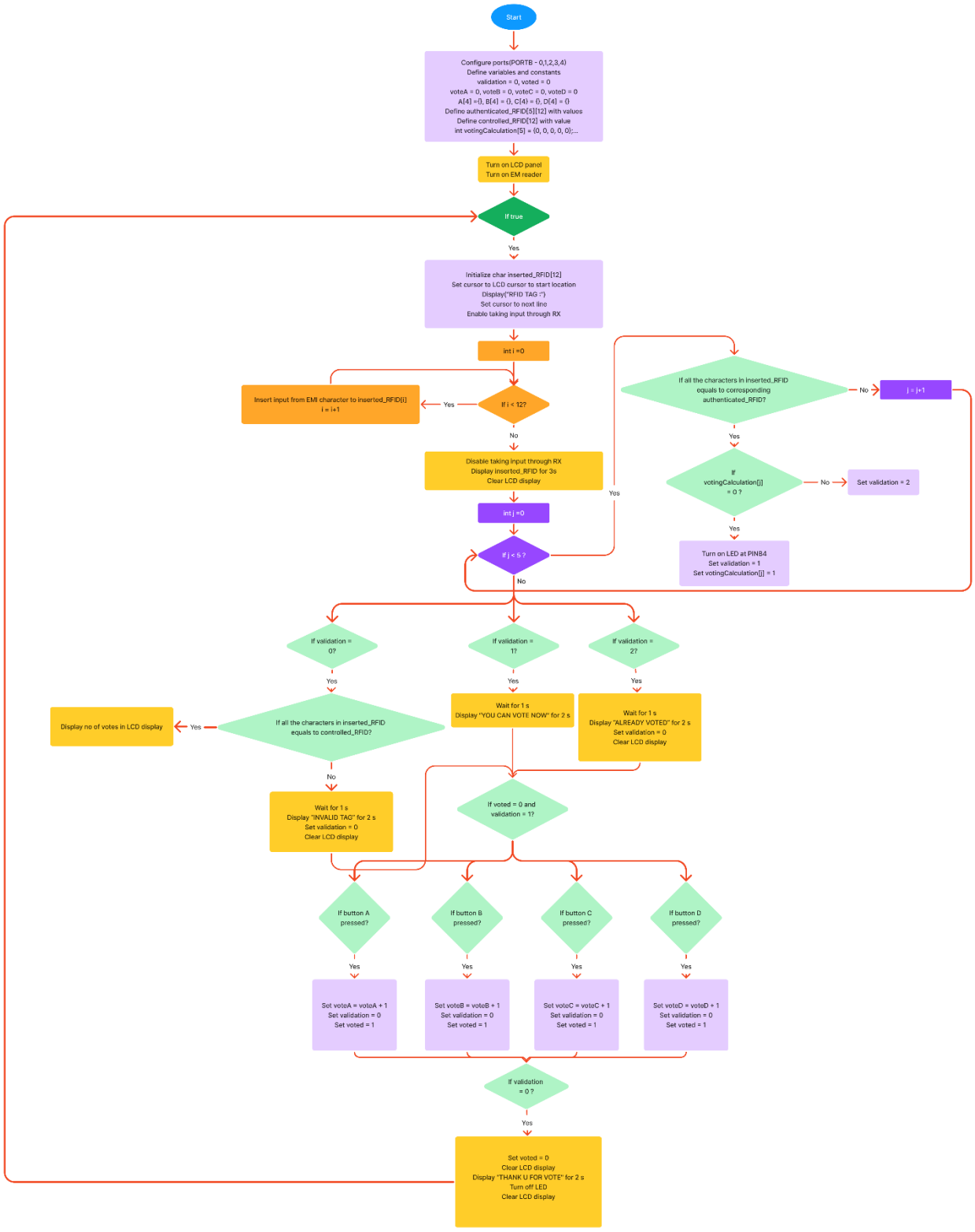

Figure 3-3 Flow Chart of the Firmware

The above diagram in Figure 3-2 describes the flow chart of the firmware. (Zoom in the diagram to see clear) it describes how the total firmware works when the system is connected to the power.

At the beginning of the process we have to define the PORTB pin 0,1,2,3 and 4 as inputs of the ATMega328p. Then integer variables are initialized as validation, voted, voteA, voteB, voteC, voted.

The variable validation is used to identify the validity of the inserted RFID tag and voted is used to check whether the current inserted RFID tag has voted or not in the current voting round. voteA, voteB, voteC and voteD variables are used to store the number of votes for the each vote A, B, C and D. Then A[4], B[4], C[4] and D[4] is used to store the converted voting numbers into characters when needed to display. Then at the beginning, system is provided valid_RFID tags they can only participate for the voting. controlled_RFID tag is defined and this tag only has the power to ending the voting process. Each RIFD tag is consist of 12 characters. Then an array with zero values is initialized to indicate that initially all the valid RFID tags are not voted(votingCalclualtions[5]).

Then when the power is given to the system, in clear LCD display, first it displays "RFID TAG :" and waits until the RFID tag is read. Once the RFID tag is read it is saved into the inserted_RFID variable and the tag number is displayed in the LCD display in next line for 3s.

Then it checks whether the inserted_RFID tag is equal to the valid_RFID tags defiend earlier through for loop. If it is valid one then first it checks whether it has previously voted or not by checking the votingCalculations array. If it is a previously not voted one, then validation is set to 1 and relevant votingCalculation index is set to 1. If it is a previously voted one, then validation is set to 2. Once RFID tag is found in valid_RFID tag array after assigning this value, if loop is broken.

Then validation may have one of these values: 0,1 or 2. Then through if loop validation value is checked. If validation is still 0, then the inserted_RFID tag is checked with the controlled_RFID. If it is equal to the controlled_RFID voting process is ended and display the voting results forever because of the break statement until the ATMega328p is reset. Otherwise LCD display displays "INVALID TAG" for 2s and LCD display is cleared and got to the start of while loop. If the validation is equal to 1 then it waits for 1s and displays "YOU CAN VOTE NOW" in LCD display continuously until the person is voted. If validation is equal to 2 then it waits for 1s and displays "ALREADY VOTED" in LCD display.

After that, in a while loop is voted is equal 0 and validation is 1, below processes are going on. It waits until a voter presses one button. Once a button is pressed then, relevant voting number is increased by one and validation is set to zero and voted set to 1.

Then inside the same loop it checked the validation value. This is used to define the processes that should be done after a vote is given. Then once validation is set to zero again voting is set to zero and LCD display is cleared and display "THANK UFOR VOTE" for 2s and LED bulb is turned off and LCD display is cleared.

3.4 REQUIRED EQUATIONS, CALCULATIONS AND EXPLANATION

When considering the project here, we have to configure the RX port for EM18 reader module functionality. In our project, when building the code, separate header file is used for its configurations. [2]

First the BAUD rate should be set using UBRR register. Value of the UBRR register is calculated using the following equation.

$$UBRR = \frac{f_{osc}}{16BAUD} - 1$$

UBRR register is a 16 bit register and higher 8 bits are UBRR0H and lower 8 bits UBRR0L.

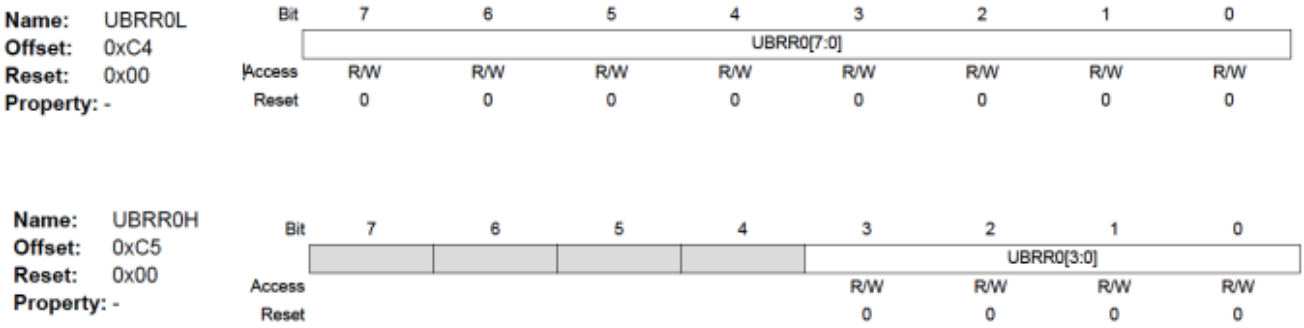

Figure 3-4 UBRR0H and UBRR0L Register

The above Figure 3-3 describes the register configuration of UBRR0H and UBRR0L. As the EM18 reader module sends the data to controller with a baud rate of 9600bps first baud rate and using the above mentioned equation, prescale is defined in the hex file as a equation as follows:

```
#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)
```

Then inside the initializing function first 8 bits is set to the upper register and remain is set to the lower register by using the following code.

```
// Set baud rate
UBRR0H = (uint8_t)(BAUD_PRESCALE >> 8);
UBRR0L = (uint8_t)(BAUD_PRESCALE);
```

Further when initializing the UART, we have to initialize the frame as well. To initialize the frame format.

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

Figure 3-5 UCSZ0 Bit Configuration

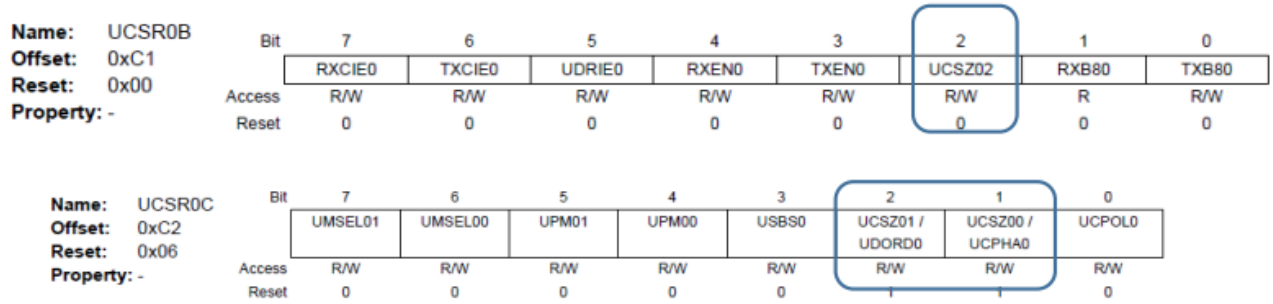

Figure 3-6 UCSZ0 Locations in the Registers

The above Figure 3-4 and 3-5 describes the bits using for setting character size and how they should be set. In our case we have to use 8-bit.

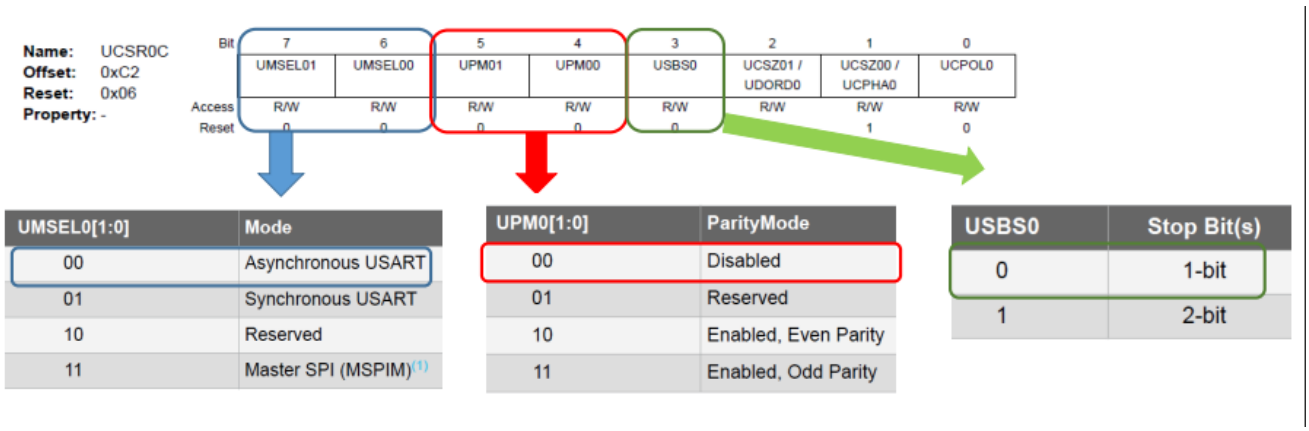

Figure 3-7 Configuring the Frame

The data in Figure 3-6 is used to further configure the frame. For the project as we are using the asynchronous mode, we have to set up it and we used parity bit disabled and 1 stop bit. Then we configure UCSR0C register as follows.

```
// Set frame format: 8 data bits, 1 stop bit, no parity
UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
```

When initializing the UART configuration we have to enable receiver port.

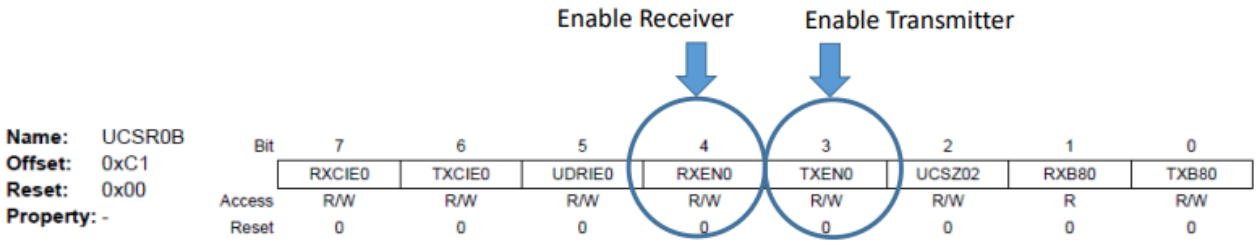

Figure 3-8 UCSR0B Register

The above Figure 3-4 represents the UCSR0B register. As we have to enable only the receiver here we have to enable RXEN0 bit. But in our system, EM18 reader should read the input only when needed. So we have to create two functions to enable and disable RX port. It has been initialized in the enableRX() function using the following code.

```
UCSR0B |= (1 << RXEN0);
```

DisableRX() has been built using the same bit of the register.

```
UCSR0B &= ~(1 << RXEN0);
```

3.5 FINAL PRODUCT

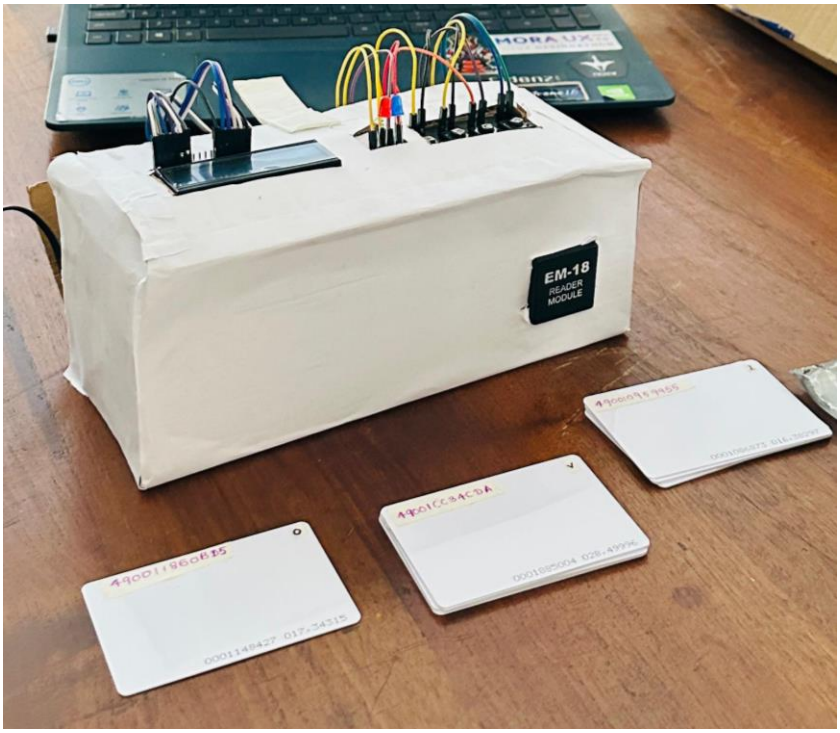

Figure 3-9 Image of the Final Product

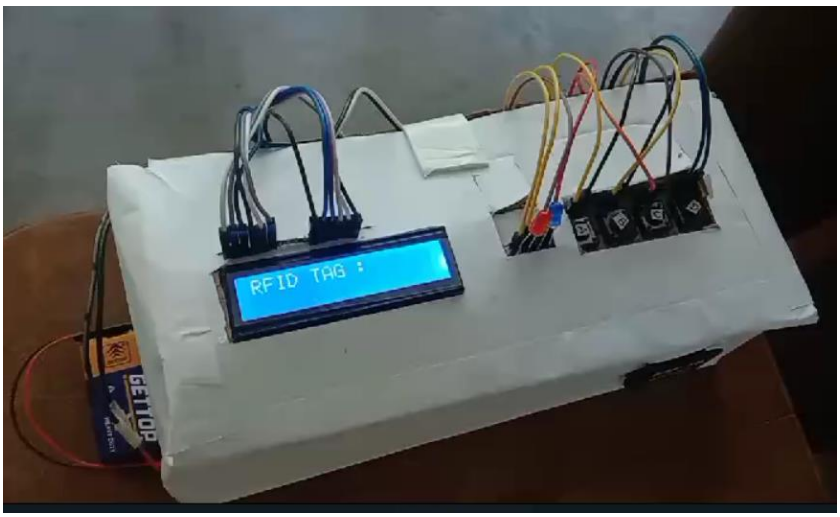

Figure 3-10 Image of the Final Product which shows the Top Side

The above Figure 3-9 and 3-10 shows the views of the final product built for the authenticated voting system.

4 REFERENCES

[1] Wikipedia, "Wikipedia," Wikipedia, 15 May 2024. [Online]. Available: https://en.wikipedia.org/wiki/Electronic_voting_machine. [Accessed 23 June 2024].

[2] DatasheetsPDF.com, "DatasheetsPDF.com," DatasheetsPDF.com, 2006. [Online]. Available: https://datasheetspdf.com/datasheet-pdf/1469778/ATmega328P.html. [Accessed 26 June 2024].