MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report

of

Embedded Systems (CSE2223)

Curtain Master

Team Details:

SECTION- CSE B

Disha Jain	220905554
Snehansh Jerath	220905486
Naman Chaubey	220905282
Chiranth Reddy	220905185
Adel Sarah Dsouza	220905300

CURTAIN MASTER

The "CurtainMaster" project is an LPC1768-based automated curtain opener system designed to provide convenience by automating the opening and closing of curtains or blinds using a stepper motor.

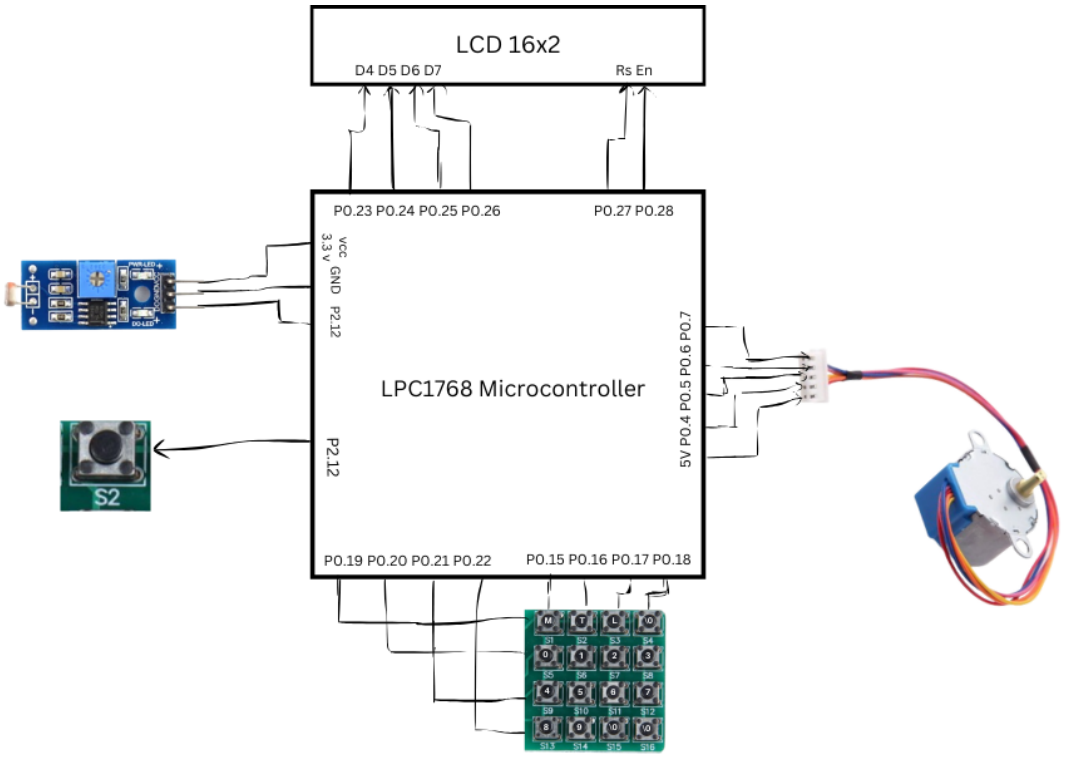
Hardware Requirements:

1. LPC1768 Microcontroller
2. Stepper Motor
3. Light Sensor
4. Switches
5. LCD Display
6. Keypad

Software Requirements:

1. Keil microvision4 simulator
2. Flash Magic

Block Diagram



Methodology:

Hardware Design:

- Design the circuitry to connect the LPC1768 microcontroller with the stepper motor, light sensor, switches, LCD display, and keypad.
- Select appropriate components and ensure compatibility with the LPC1768.

Firmware Development:

- Set up the Keil uVision4 integrated development environment (IDE) for LPC1768 development.
- Write firmware code in C to implement the required functionalities:
    - Control the stepper motor to open and close the curtains.
    - Read input from the light sensor to automate curtain operation based on ambient light levels.
    - Handle user input from the keypad to adjust settings and control manual operation.
    - Display relevant information on the LCD display.
    - Implement safety features and error handling mechanisms to ensure reliable operation.
- Test the firmware code using the Keil uVision4 simulator to verify correctness and functionality.

Software Integration:

- Integrate the firmware code with the hardware components, ensuring proper communication and functionality.
- Verify that the firmware interacts correctly with each hardware component and responds appropriately to inputs and sensor readings.

Testing and Debugging:

- Conduct thorough testing of the integrated system to identify and resolve any bugs or issues.
- Test various scenarios, including automatic operation based on light levels, manual operation via switches, and user input via the keypad.
- Debug any errors or unexpected behaviour encountered during testing.
- Ensure the system operates reliably and meets the specified requirements.

Deployment:

- Use Flash Magic or a similar tool to program the compiled firmware code onto the LPC1768 microcontroller.
- Assemble the hardware components into the final system configuration, ensuring proper connections and mounting.

Limitations:

1. **Non-Concurrent Operations:** One of the limitations of the CurtainMaster system is the inability to perform multiple operations simultaneously. Due to hardware constraints or design limitations, the system can only execute one task at a time. This restriction affects the system's responsiveness and may introduce delays in operation, particularly when transitioning between tasks.

2. **Time Constraint:** Another limitation stems from the complexity of providing multiple inputs within a limited timeframe. The system can only count time intervals accurately up to 9 minutes. This limitation arises from the challenge of accommodating multiple user inputs within the allotted time frame, which may impact user experience for tasks requiring precise timing or scheduling.

3. **Inoperable Light Sensor:** Due to compatibility issues or hardware limitations with the selected kit, the light sensor functionality could not be implemented as intended.

4. **Curtain Overflow:** A critical limitation identified during testing is the potential for curtain overflow. The system may encounter issues when attempting to close or open the curtains beyond their physical limits. This can result in motor strain, component damage, or disruption of curtain operation. Addressing this limitation is crucial to ensure the safety and longevity of the system.

Functionalities:

Manual Timer Functionality: The manual timer functionality in the provided code allows users to manually control the opening and closing of curtains using a keypad and a switch. After initializing the system and waiting for user input, the program enters manual mode when the 'M' key is pressed. In manual mode, the user can use the switch to rotate the stepper motor either clockwise or counterclockwise to open or close the curtains, respectively. Manual operation continues until the 'M' key is pressed again to exit manual mode.

```
void manual(void) {
    while (1) {
                key=0;
                var1 = 1 << 19;
        temp = var1;
        LPC_GPIO0->FIOCLR = 0xF << 19;
        LPC_GPIO0->FIOSET = var1;
        flag = 0;
        scan();
        if (key == 0x11)
            break;   //the above code is to stop the curtain from rotating if M is pressed again
        x = LPC_GPIO2->FIOPIN >> 12;
        x &= 1;
        if (x == 1)
            clockwise();
        else if (x == 0)
            anti_clockwise();
                //figure out how to close
    }
}
```

Timer Functionality: the program enters timer mode when the 'T' key is pressed The timer functionality implemented in the provided code allows users to set a timer for a specific duration, after which the curtains will automatically open or close. Users can input the desired duration using the keypad, and the program converts the input into minutes. Once the timer is set, the program utilizes a match timer interrupt to trigger the rotation of the stepper motor after the specified duration elapses. This automation provides users with the convenience of scheduling curtain operations without manual intervention. However, it's important to note that the timer functionality is subject to a maximum duration of 9 minutes due to constraints in processing multiple inputs within a limited timeframe.

```
void settime() {

    clear_ports();
    delay(milliseconds: 30);

    lcd_comdata(temp1: 0xC0, type: 0); // point to first line of LCD
    delay(milliseconds: 800);
    while (1) {
        for (row = 2; row < 5; row++) {
            if (row == 2)
                var1 = 1 << 20;
            else if (row == 3)
                var1 = 1 << 21;
            else if (row == 4)
                var1 = 1 << 22;
            temp = var1;
            LPC_GPIO0->FIOCLR = 0x00780000; // first clear the
            LPC_GPIO0->FIOSET = var1;        // enabling the row
            flag = 0;
            scan(); // scan if any key pressed in the enabled row
            if (flag == 1)
                break;
        } // end for
        if (flag == 1)
            break;
    } // 2nd while(1)
    for (i = 4; i < 16; i++) // get the ascii code for display
    {
        if (key == SCAN_CODE[i]) {
            key = ASCII_CODE[i];
            break;
        } // end if(key == SCAN_CODE[i])
    }     // end for(i=0;i<16;i++)
```

```
    lcd_puts(buf1: &key);
    key = key - 48;
    time = key * 60 * 1000;

    Match(milliseconds: time);
}
```

Light Functionality: the program enters manual mode when the 'L' key is pressed. The light functionality is designed to automatically control the curtains based on ambient light levels sensed by a light sensor. After initializing the ADC and configuring pin P1.31 as an ADC input, the program reads the ADC value to determine the ambient light level. If the light level exceeds a predefined threshold, the program rotates the stepper motor counterclockwise to close the curtains, assuming it's daytime. However, due to potential limitations or constraints, the effectiveness of this functionality may vary, and further refinement may be necessary for optimal automation.

```
void light(void) {
    LPC_SC->PCONP |= (1 << 12); //give power supply to ADC
    LPC_PINCON->PINSEL3 |= 0xC0000000; // P1.31 as AD0.5 fn3

    LPC_ADC->ADCR = (1 << 5) | (1 << 21) | (1 << 24); // 0x01200001;
                                                      // ADC0.5, start conversion and operational
    while (!(LPC_ADC->ADDR5 & 0x80000000));
    // wait till 'done' bit is 1, indicates conversion complete

    adc_temp = LPC_ADC->ADDR5;
    adc_temp >>= 4;
    adc_temp &= 0x00000FFF; // 12 bit ADC
    in_vtg = (((float)adc_temp * (float)Ref_Vtg)) / ((float)Full_Scale);
    // calculating input analog voltage
    delay(milliseconds: 300);
    if (in_vtg > 0.7) {
        for (i = 0; i < 400; i++)
            anti_clockwise();
    }
    // if this in_vtg is above a certain value then close blinds
    adc_temp = 0;
    in_vtg = 0;
}
```

Code:

```
#include <lpc17xx.h>

unsigned int v1, x, i;
unsigned char row, var, flag, key;
unsigned long int var1, temp, temp1, temp2, temp3, time;
unsigned char SCAN_CODE[16] = {0x11, 0x21, 0x41, 0x81,
                               0x12, 0x22, 0x42, 0x82,
                               0x14, 0x24, 0x44, 0x84,
                               0x18, 0x28, 0x48, 0x88};
unsigned char ASCII_CODE[16] = {'M', 'T', 'L', '\0',
                                '0', '1', '2', '3',
                                '4', '5', '6', '7',
                                '8', '9', '\0', '\0'};
#define Ref_Vtg 3.300
#define Full_Scale 0xFFF // 12 bit ADC
unsigned long adc_temp;
float in_vtg;

void lcd_init(void);
void write(int, int);
void delay(unsigned int);
void lcd_comdata(int, int);
void clear_ports(void);
void lcd_puts(unsigned char *);

void lcd_init() {
    /*Ports initialized as GPIO */
            LPC_PINCON->PINSEL1 &= 0xFC003FFF; //P0.23 to P0.28
            /*Setting the directions as output */
            LPC_GPIO0->FIODIR |= 0x0F<<23 | 1<<27 | 1<<28;
            clear_ports();
            delay(200);
            lcd_comdata(0x33, 0);
            delay(300);
            lcd_comdata(0x32, 0);
            delay(300);
            lcd_comdata(0x28, 0); //function set
            delay(300);
            lcd_comdata(0x0c, 0);//display on cursor off
            delay(800);
            lcd_comdata(0x06, 0); //entry mode set increment cursor right
            delay(800);
            lcd_comdata(0x01, 0); //display clear
            delay(1000);
            return;
}
```

```
void lcd_comdata(int temp1, int type) {
    int temp2 = temp1 & 0xf0; // move data (26-8+1) times : 26 - HN place, 4 - Bits
    temp2 = temp2 << 19;      // data lines from 23 to 26
    write(temp2, type);
    temp2 = temp1 & 0x0f; // 26-4+1
    temp2 = temp2 << 23;
    write(temp2, type);
    delay(1000);
    return;
}

void write(int temp2, int type) { /* write to command/data reg */
    clear_ports();
    LPC_GPIO0->FIOPIN = temp2; // Assign the value to the data lines
    if (type == 0)
        LPC_GPIO0->FIOCLR = 1 << 27; // clear bit RS for Command
    else
        LPC_GPIO0->FIOSET = 1 << 27; // set bit RS for Data
    LPC_GPIO0->FIOSET = 1 << 28;    // EN=1
    delay(25);
    LPC_GPIO0->FIOCLR = 1 << 28; // EN =0
    return;
}

void inittime0() {
    LPC_TIM0->CTCR = 0x0;
    LPC_TIM0->TCR = 0x2;
    LPC_TIM0->PR = 2999; //Pclk=3MHz and Tres = 1ms
}

void delay(unsigned int milliseconds) {
    LPC_TIM0->TCR = 0x2;
    LPC_TIM0->TCR = 0x1;
    while (LPC_TIM0->TC < milliseconds);
    LPC_TIM0->TCR = 0x0;
}

void clockwise() {
    v1 = 0x8;
    for (i = 0; i < 4; i++) {
        v1 = v1 << 1;
        LPC_GPIO0->FIOPIN = v1;
        delay(10000);
    }
}

void anti_clockwise() {
    v1 = 0x100;
```

```
    for (i = 0; i < 4; i++) {
        v1 = v1 >> 1;
        LPC_GPIO0->FIOPIN = v1;
        delay(10000);
    }
}

void Match(int milliseconds) {
    LPC_TIM0->MR0 = milliseconds;
    LPC_TIM0->MCR = 2;
    LPC_TIM0->EMR = 0x20;
    LPC_TIM0->TCR = 0x2;
    LPC_TIM0->TCR = 0x1;
    while (!(LPC_TIM0->EMR & 1));
    LPC_TIM0->TCR = 0x0;
    for (i = 0; i < 400; i++)
        clockwise();
}

void clear_ports(void) { /* Clearing the lines at power on */
    LPC_GPIO0->FIOCLR = 0x0F << 23; // Clearing data lines
    LPC_GPIO0->FIOCLR = 1 << 27;    // Clearing RS line
    LPC_GPIO0->FIOCLR = 1 << 28;    // Clearing Enable line
    return;
}
void scan(void) {
    temp3 = LPC_GPIO0->FIOPIN;
    temp3 &= (0XF << 15);
    if (temp3 != 0x00000000) {
        flag = 1;
        temp3 >>= 11; // 15-4
        temp >>= 19;
        key = temp3 | temp;
    }
}

void manual(void) {
    while (1) {
              key=0;
              var1 = 1 << 19;
        temp = var1;
        LPC_GPIO0->FIOCLR = 0xF << 19;
        LPC_GPIO0->FIOSET = var1;
        flag = 0;
        scan();
        if (key == 0x11)
            break;   //the above code is to stop the curtain from rotating if M is
pressed again
        x = LPC_GPIO2->FIOPIN >> 12;
```

```
        x &= 1;
        if (x == 1)
            clockwise();
        else if (x == 0)
            anti_clockwise();
                //figure out how to close
    }
}

void settime() {

    clear_ports();
    delay(30);

    lcd_comdata(0xC0, 0); // point to first line of LCD
    delay(800);
    while (1) {
        for (row = 2; row < 5; row++) {
            if (row == 2)
                var1 = 1 << 20;
            else if (row == 3)
                var1 = 1 << 21;
            else if (row == 4)
                var1 = 1 << 22;
            temp = var1;
            LPC_GPIO0->FIOCLR = 0x00780000; // first clear the
            LPC_GPIO0->FIOSET = var1;        // enabling the row
            flag = 0;
            scan(); // scan if any key pressed in the enabled row
            if (flag == 1)
                break;
        } // end for
        if (flag == 1)
            break;
    } // 2nd while(1)
    for (i = 4; i < 16; i++) // get the ascii code for display
    {
        if (key == SCAN_CODE[i]) {
            key = ASCII_CODE[i];
            break;
        } // end if(key == SCAN_CODE[i])
    }     // end for(i=0;i<16;i++)
    lcd_puts(&key);
    key = key - 48;
    time = key * 60 * 1000;

    Match(time);
}
```

```
void lcd_puts(unsigned char *buf1) {
    unsigned int i = 0;
    unsigned int temp3;
    while (buf1[i] != '\0') {
        temp3 = buf1[i];
        lcd_comdata(temp3, 1);
        i++;
        if (i == 16) {
            lcd_comdata(0xc0, 0);
        }
    }
    return;
}

void light(void) {
    LPC_SC->PCONP |= (1 << 12); //give power supply to ADC
    LPC_PINCON->PINSEL3 |= 0xC0000000; // P1.31 as AD0.5 fn3

    LPC_ADC->ADCR = (1 << 5) | (1 << 21) | (1 << 24); // 0x01200001;
                                                      // ADC0.5, start conversion and
operational
    while (!(LPC_ADC->ADDR5 & 0x80000000));
    // wait till 'done' bit is 1, indicates conversion complete

    adc_temp = LPC_ADC->ADDR5;
    adc_temp >>= 4;
    adc_temp &= 0x00000FFF; // 12 bit ADC
    in_vtg = (((float)adc_temp * (float)Ref_Vtg)) / ((float)Full_Scale);
    // calculating input analog voltage
    delay(300);
    if (in_vtg > 0.7) {
        for (i = 0; i < 400; i++)
            anti_clockwise();
    }
    // if this in_vtg is above a certain value then close blinds
    adc_temp = 0;
    in_vtg = 0;
}
unsigned char a[]="Start";
int main(void) {
      SystemInit();
      SystemCoreClockUpdate();
    LPC_PINCON->PINSEL0 = 0X0; // p0.4 to p0.7 stepper motor
    LPC_PINCON->PINSEL4 = 0X0; // SWITCH P2.12 manual
    LPC_GPIO2->FIODIR = 0 << 12;
    LPC_GPIO0->FIODIR |= 0xf << 4 | 0xf << 19 | 0x0 << 15; // made output P0.19 to P0.22
(rows)
                                                          // made input P0.15 to
P0.18(cols)
```

```
    inittime0();
      lcd_init();

      lcd_puts(&a[0]);
    while (1) {
        var1 = 1 << 19;
        temp = var1;
        LPC_GPIO0->FIOCLR = 0xF << 19;
        LPC_GPIO0->FIOSET = var1;
        flag = 0;
        scan();
        if (flag == 1)
            break;
    }
    for (i = 0; i < 4; i++) {
        if (key == SCAN_CODE[i]) {
            key = ASCII_CODE[i];
            break;
        }
    }

    switch (key) {
    case 'M':
        manual();
        break;

    case 'T':
        settime();
        break;
    case 'L':
        light();
        break;
    default:
        break;
    }

}
```

Conclusion:

In conclusion, the "CurtainMaster" project represents a significant step towards enhancing convenience and efficiency in curtain management. Despite facing challenges such as hardware limitations and time constraints, the system delivers essential functionalities, including manual timer control and light sensing capabilities. While further refinement is needed to address constraints and optimize performance, the project underscores the potential for smart home automation, offering users intuitive control over curtain operations and laying the groundwork for future advancements in user-centric design and automation technology.

References:

http://www.ocfreaks.com/interfacing-ldr-lpc1768/

https://www.keil.com/dd/docs/datashts/philips/lpc17xx_um.pdf