**CAR PARKING MANAGEMENT SYSTEM**

# TERM PROJECT

***Submitted by***

# CB.EN.U4CCE21003 ABISHEK S

**CB.EN.U4CCE21016 DHIVYADARSHAN G**

# CB.EN.U4CCE21045 PIRAI SOODAN G

# CB.EN.U4CCE21067 SIDARTH M



**B.TECH COMPUTER AND COMMUNICATION ENGINEERING**

# 2021 BATCH

TABLE OF CONTENTS

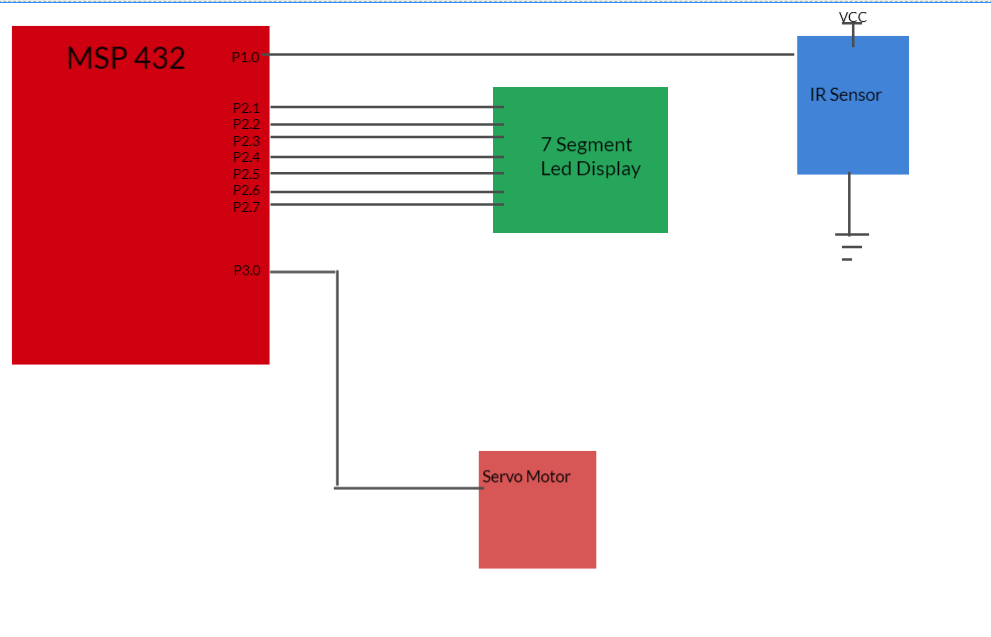
1. Aim
2. Design
3. Peripherals and its pins
4. Code
5. Implementation Status
6. Inference
7. Conclusion

Aim:

To implement a simple car parking management system using MSP432.

* We have implemented a car parking management system where all the parking slots are fitted with IR Sensors, each of which will detect the presence of car in their respective slots and send the acquired data to MSP432 and the number of the parking slots available/free will be shown in Hyperterminal as well as the 7 segment led display. When the available parking slots are 0, the servo motor will close the gate and when it changes to 1 or 2, the servo motor will open the gate.
* Components Required:
* MSP 432
* Servo Motor
* IR Sensors
* 7 Segment Display
* UART

Design:



UART

Peripherals and its pins:

|  |  |  |
| --- | --- | --- |
| **Peripherals** | **Pins** | **Description** |
| MSP432 |  | MSP432 is a microcontroller which follows ARM v6. |
| Universal Asynchronous Rx/Tx (UART0) | P1.2 , P1.3 | It is used to transmit and receive data. |
| Servo Motor | P3.0 | Used to close the gate when a car is detected and open otherwise |
| IR Sensor | P1.0 | Used to detect the presence/absence of a car |
| 7 Segment LED Display | P2.0 to p2.7 | Used to display the number of free parking slots |

Code

#include "msp.h"

#include <stdio.h>

#include <stdbool.h>

#define IR\_SENSOR\_PIN BIT5

#define IR\_SENSOR2\_PIN BIT4

#define SERVO\_CONTROL\_PIN BIT0

#define LED\_PIN BIT0

#define SERVO\_PIN BIT7

#define PARKING\_FULL\_PIN BIT0

void configureIRSensor();

void configureServoMotor();

void delayMs(int n);

void servoRotateClockwise();

void servoRotateCounterClockwise();

void UART\_Init(void);

void UART\_TransmitChar(unsigned char data);

void UART\_TransmitString(const char\* str);

void itoa(int value, char\* str, int base);

void configureSevenSegment();

void displayNumber(int num);

int availableSpaces = 2; // Set initial available spaces

char str[3];

char report[50];

void configureIRSensor() {

// Configure IR sensor pins as inputs with pull-up resistors

P1->DIR &= ~IR\_SENSOR\_PIN;

P1->REN |= IR\_SENSOR\_PIN;

P1->OUT |= IR\_SENSOR\_PIN;

P4->DIR &= ~IR\_SENSOR2\_PIN;

P4->REN |= IR\_SENSOR2\_PIN;

P4->OUT |= IR\_SENSOR2\_PIN;

}

void configureServoMotor() {

// Configure servo motor control pin as output

P3->DIR |= SERVO\_CONTROL\_PIN;

}

void configureSevenSegment() {

// Configure GPIO pins for 7-segment LED display

// Adjust the pin assignments based on your specific hardware setup

P2->DIR |= BIT0 | BIT1 | BIT2 | BIT3 | BIT4 | BIT5 | BIT6;

}

void displayNumber(int num) {

// Define the 7-segment LED display patterns for digits 0-9

uint8\_t patterns[10] = {

0x3F, // 0

0x06, // 1

0x5B, // 2

0x4F, // 3

0x66, // 4

0x6D, // 5

0x7D, // 6

0x07, // 7

0x7F, // 8

0x6F // 9

};

// Display the specified number on the 7-segment LED

P2->OUT = patterns[num % 10];

}

void delayMs(int n) {

int i, j;

for (j = 0; j < n; j++)

for (i = 750; i > 0; i--); // Delay of 1 ms

}

void servoRotateClockwise() {

// Rotate the servo motor clockwise by setting the control pin high for a specific duration

P3->OUT |= SERVO\_CONTROL\_PIN;

delayMs(2); // Adjust the delay to control the rotation angle

P3->OUT &= ~SERVO\_CONTROL\_PIN;

}

void servoRotateCounterClockwise() {

// Rotate the servo motor counter-clockwise by setting the control pin high for a specific duration

P3->OUT |= SERVO\_CONTROL\_PIN;

delayMs(1); // Adjust the delay to control the rotation angle

P3->OUT &= ~SERVO\_CONTROL\_PIN;

}

void UART\_Init(void) {

EUSCI\_A0->CTLW0 |= 1; // Put in reset mode to configure UART

EUSCI\_A0->MCTLW = 0; // Disable oversampling

EUSCI\_A0->CTLW0 = 0x0081; // 00 - 1 stop bit, No Parity, 8-bit data, Asynchronous Mode, First LSB Then MSB, SMCLK

// 81 - Enabled EUSCI\_A0 logic held in reset state

EUSCI\_A0->BRW = 26; // 3000000/115200 = 26

P1->SEL0 |= 0x0C; // Configure functionality of P1.2, P1.3 as UART Pins

P1->SEL1 &= ~0x0C;

EUSCI\_A0->CTLW0 &= ~1; // Take UART out of reset mode

}

void UART\_TransmitChar(unsigned char data) {

while (!(EUSCI\_A0->IFG & 0x02)); // Wait until TX buffer is ready

EUSCI\_A0->TXBUF = data; // Send character

}

void UART\_TransmitString(const char\* str) {

while (\*str) {

UART\_TransmitChar(\*str);

str++;

}

}

void itoa(int value, char\* str, int base) {

sprintf(str, "%d", value);

}

int main(void) {

\_\_disable\_irq();

// Configure IR sensor pins with pull-up resistors

configureIRSensor();

// Configure servo motor control pin as output

configureServoMotor();

// Configure LED output pin

P1->DIR |= LED\_PIN;

// Initialize the 7-segment display

configureSevenSegment();

UART\_Init();

// Enable interrupts

\_\_enable\_irq();

while (1) {

if (((P1->IN & IR\_SENSOR\_PIN) == 0) && ((P4->IN & IR\_SENSOR2\_PIN) == 0)) {

// Both IR sensors detect an object

P1->OUT ^= LED\_PIN; // Toggle LED state

servoRotateClockwise();

availableSpaces = 0;

delayMs(500);

} else if (((P1->IN & IR\_SENSOR\_PIN) == 0) || ((P4->IN & IR\_SENSOR2\_PIN) == 0)) {

// Either one of the IR sensors detects an object

P1->OUT &= ~LED\_PIN; // Turn off LED

servoRotateCounterClockwise();

availableSpaces = 1;

delayMs(500);

} else {

// Neither IR sensor detects an object

P1->OUT &= ~LED\_PIN; // Turn off LED

servoRotateCounterClockwise();

availableSpaces = 2;

delayMs(500);

}

// Update 7-segment display with available spaces

displayNumber(availableSpaces);

// Transmit report over UART

sprintf(report, "Available spaces: %d\n", availableSpaces);

UART\_TransmitString(report);

delayMs(2000);

}

}

Implementation Status:

We’ve configured the IR Sensor and Servo motor correctly. So when a car is sensed by IR Sensor, The Servo motor will rotate clockwise(Gate is opened). Then after the car gets inside and nothing is sensed by the IR Sensor, the gate will rotate anticlockwise(Gate is closed). Also the status of the parking slot can be seen in the Hyperterminal i.e “Parking free” when nothing is sensed and “Parking full” when all slots are occupied. We’ve also been able to print the number of available parking slots in 7 segment display.

Inference:

A car parking management system based on the MSP432 microcontroller offers numerous advantages. It enhances parking efficiency by automating processes and optimizing space utilization. Real-time spot availability information allows drivers to make informed decisions, reducing search times and improving overall user satisfaction.

With streamlined operations, reduced wait times, and enhanced convenience, the car parking management system improves the overall parking experience while maximizing the efficiency and effectiveness of parking facilities

Conclusion:

In conclusion, the implementation of a car parking management system using the MSP432 microcontroller streamlines operations, improves the parking experience, and maximizes the effectiveness of parking facilities, ultimately leading to increased user satisfaction and improved resource allocation. By leveraging the capabilities of the MSP432, the car parking management system offers a comprehensive solution for managing parking spaces efficiently and effectively.

Scope for future use:

This model can be scaled to a large scale implementation such as parking management system for malls, theater where we can divide the already present parking lots into smaller sectors and generate report for each individual sectors by doing which we can identify which sectors are empty and which of them are full, thereby when a user comes to park the vehicle, they can be routed to empty sector which will reduce the waste of time that will normally happen in search of free slots.