Project: Smart Parking System

Course: CPEN 305 - Microcontrollers Lab

Prepared by: Melissa Naffah & Hassan Mroueh

Instructor: Ms. Katia Karam

Date: 9-5-2024

1. **Introduction:**

The necessity for effective and smart parking solutions has grown in this era of fast urbanization and rising vehicle density. The difficulty of effectively managing parking spaces increases with the size of cities, which frequently results in traffic jams, annoyance, and lost time for drivers.   
One revolutionary approach to addressing these issues is the use of Internet of Things (IoT) technology into parking systems. IoT-enabled smart parking systems increase parking space use, improve user experience, and reduce traffic congestion by utilizing sensors, connectivity, and data analytics.

Through the design and installation of a smart parking system, this project seeks to meet the parking needs of a tiny parking lot with three parking spaces. This technology will transform parking by automating procedures, giving real-time spot availability information, and increasing overall efficiency by utilizing the power of the Internet of Things.

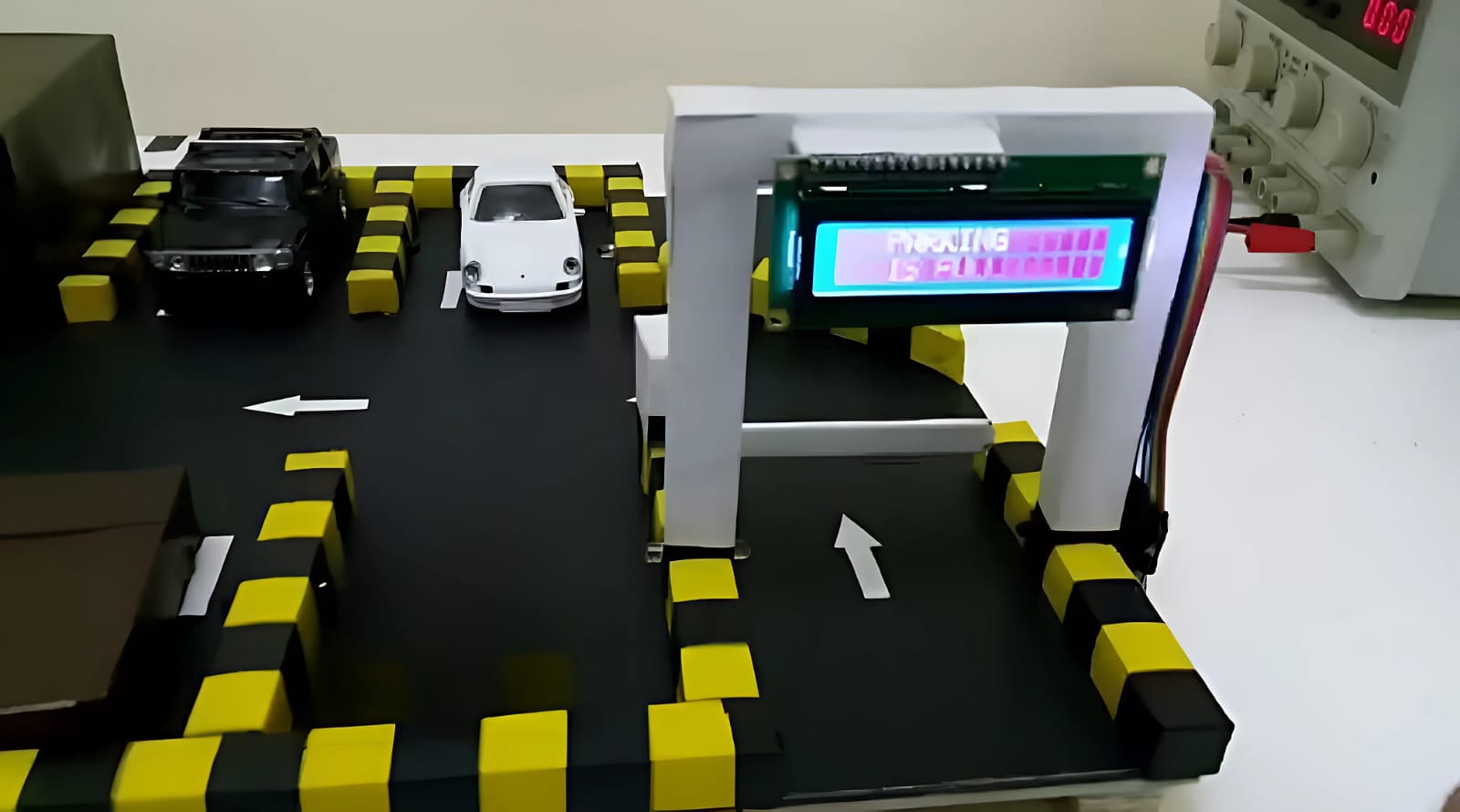


Figure 1: Schema of the Smart Parking System

We have three things that we must take into consideration:

**Car Gate Control:** When a car approaches or leaves the property, an intelligent gate system will be put in place to automatically open and close. Users will find the parking process more efficient and convenient because of this automated approach, which removes the need for manual intervention.

**Real-time Spot Availability Display:** To continuously show the availability of parking spots in real-time, a sizable LCD display will be erected at the parking lot's entrance. With the help of this tool, drivers may plan their parking more intelligently, cutting down on time wasted looking for a spot and easing traffic at the entry.

**Integration of Sensors:** To detect cars coming into and going out of the parking lot, proximity sensors will be smoothly integrated. With the use of these sensors, the system will be able to precisely monitor the occupancy status of parking spaces in real-time, facilitating the effective distribution of available spaces and the dynamic management of parking resources.

**Main components:**

* **Microcontroller unit: PIC18F45K22** which was used in all the labs before.
* **Servo motor:** a servo motor would be a good option to operate the gate mechanism in this intelligent parking system. Because they can precisely regulate an angular position, servo motors are perfect for applications that need precise movement control, such opening and closing gates.



* **E18-D80NK adjustable infrared proximity sensors:**

is a popular sensor used in various applications requiring detection of objects within a certain range. An object that is inside the detecting range is indicated by a digital output signal that the sensor produces. Upon detecting an object inside the designated range, the sensor's output increases to a high logic level. On the other hand, the output stays low (logic level 0) in the event that no object is found inside the range. It also operates within a specified voltage range, commonly 3.3V to 5V. This makes it compatible with a wide range of microcontrollers and embedded systems.



* **LCD display:** To let the user know how many parking spots are left and if there are no more places left, the user can’t enter the parking lot

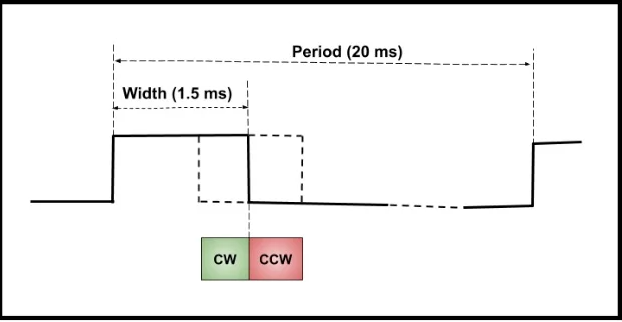
**2. Explanation of the code:**

**PWM:**

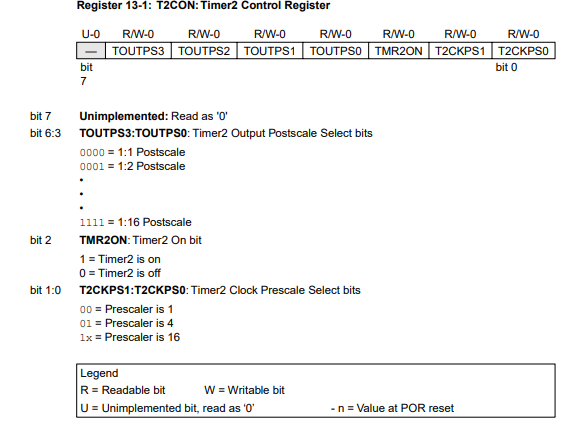
* **Definition:**

In electronics, a modulation technique called pulse width modulation, or PWM, is used to encode data in a pulsed signal. It entails altering a periodic signal's pulse width (duration) while maintaining a constant period. PWM is widely utilized in many different applications, such as communication systems, motor control, power regulation, and lighting control.   
The duty cycle in PWM is the ratio of the pulse width to the signal's overall period. It indicates the percentage of the entire period that the signal is in the high state, or on, and is commonly given as a percentage. One can regulate the effective voltage supplied to a component or the average power given to a load by varying the duty cycle.

A servo motor, which changes the angle of a shaft according to the width of the pulse. For example, a 1.5 millisecond pulse sets the shaft to 0°. Changing the pulse width from 1.5 milliseconds to 2 milliseconds goes from 0° to +90°.



* **PWM generation using CCP module**

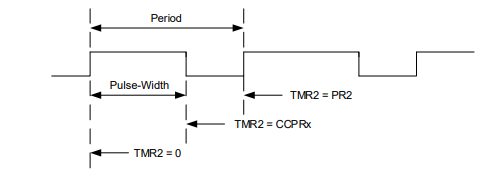


The first 4 bits of T2CON register are related to postscaler values and we’re not going to use post scaler values so they are going to be 0.

TMR2ON is used to turn timer 2 on, initially it will be 0

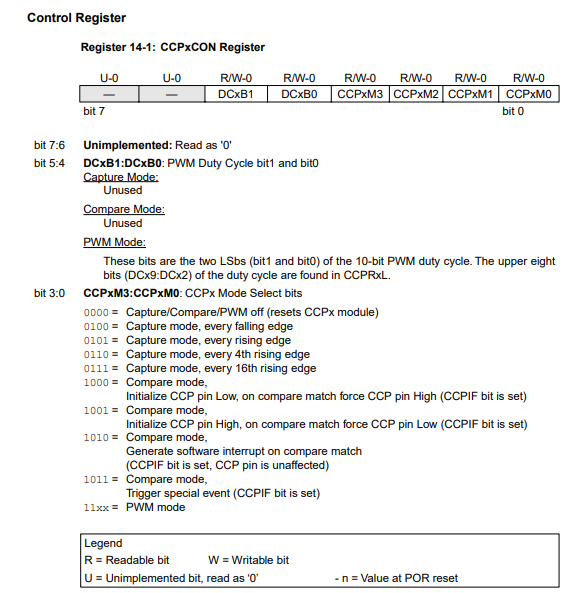
Bit 1:0 : 1x => prescaler is 16

T2CON: 00000010



A diagram of a system

Description automatically generated



Bit 3:0:

11xx since we want PWM mode

**Calculations:**

N is Prescaler (16)

Fosc= 8 MHz & Fpwm=

**Proteus code:**

1. #include <p18cxxx.h> // Include header file for the specific microcontroller being used

2. #include <LCD4lib.h> // Include library for interfacing with a 4-bit LCD

3. #include <EEPROM.h> // Include library for accessing EEPROM memory

4. #include <delays.h> // Include library for delay functions

5.

6. #define eeSeconds 0x00 // Define EEPROM address for storing Seconds variable

7. #define Device PORTCbits.RC7 // Define pin for controlling a device

8. #define Start PORTCbits.RC0 // Define pin for start button

9. #define exit PORTCbits.RC3 // Define pin for exit sensor

10. #define entrance PORTCbits.RC2 // Define pin for entrance sensor

11. #define motor PORTCbits.RC1 // Define pin for motor control

12.

13. #pragma config FOSC = INTIO67 // Configuration setting: Internal oscillator block

14. #pragma config LVP = OFF // Configuration setting: Single-Supply ICSP disabled

15. #pragma config BOREN = OFF // Configuration setting: Brown-out Reset disabled

16.

17. #define \_XTAL\_FREQ 8000000 // Oscillator frequency for delay functions

18.

19. #define plus PORTAbits.RA0 // Define pin for something named "plus"

20.

21. unsigned int ang1; // Declare variable ang1

22. int lmax, lmin; // Declare variables lmax and lmin

23. int i, servo = 1; // Declare variables i and servo, initialize servo to 1

24. char entered= 0, exited = 0; // Declare variables entered and exited, initialize to 0

25. char Seconds = 0; // Declare variable Seconds, initialize to 0 (represents number of cars)

26. char digits[3]; // Declare array to store digits

27.

28. void Setup(void); // Function prototype for Setup

29. void GateControl(void); // Function prototype for GateControl

30. void turnOn(void); // Function prototype for turnOn

31. void setup\_pwm(void); // Function prototype for setup\_pwm

32. void set\_pwm2\_duty(unsigned int duty); // Function prototype for set\_pwm2\_duty

33.

34. void main (void){ // Main function

35. lmin = 30; // Initialize lmin

36. lmax = 65; // Initialize lmax

37. ang1 = lmin; // Initialize ang1

38. while(1) { // Infinite loop

39. setup\_pwm(); // Setup PWM

40. Setup(); // Run setup function

41. turnOn(); // Run turnOn function

42. GateControl(); // Run GateControl function

43. }

44. }

45.

46. void Setup(void){ // Setup function definition

47. Seconds = 0; // Initialize Seconds to 0

48. TRISC |= 0x0D; // Set RC0, RC2, RC3 as input and RC1 as output

49. TRISCbits.RC7 = 0; // Set RC7 as output

50. ANSELC &= 0x70; // Configure RC0-RC4 as digital

51. InitLCD(); // Initialize LCD

52. DispRomStr(Ln1Ch0,(ROM)"Enter parking "); // Display message on LCD

53. DispRomStr(Ln2Ch0,(ROM)"Remaining: "); // Display message on LCD

54. ReadEE(eeSeconds,&Seconds); // Read from EEPROM to get stored value of Seconds

55. Seconds =0; // Reset Seconds to 0

56. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

57. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

58. Device =0; // Set Device pin low

59. }

60.

61. void turnOn(void){ // turnOn function definition

62. while(1){ // Infinite loop

63. if (Start == 1) { // If start button is pressed

64. Device = 1; // Set Device pin high

65. Seconds = 0; // Reset Seconds to 0

66. set\_pwm2\_duty(93); // Set PWM duty cycle

67. return; // Exit function

68. }

69. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

70. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

71. }

72. }

73.

74. void GateControl(void){ // GateControl function definition

75. while(1){ // Infinite loop

76. while(Seconds == 3){ // Wait until all spots are full

77. if (exit) { // If exit sensor is activated

78. Seconds--; // Decrement Seconds

79. while(1){ // Inner loop

80. set\_pwm2\_duty(140); // Set PWM duty cycle

81. if(entrance) // If entrance sensor is activated

82. { Delay10KTCYx(20); // Delay

83. break; // Exit loop

84. }

85. }

86. while(exit); // Wait for car to fully exit

87. }

88. }

89. if (entrance) { // If car is entering

90. while(entrance); // Wait for car to fully enter

91. while(1){ // Inner loop

92. set\_pwm2\_duty(140); // Set PWM duty cycle

93. if(exit){ // If exit sensor is activated

94. Delay10KTCYx(20); // Delay

95. break; // Exit loop

96. }

97. }

98. Seconds++; // Increment Seconds

99. while(entrance); // Wait for car to fully enter

100. } else if (exit) { // If car is exiting

101. Seconds--; // Decrement Seconds

102. while(1){ // Inner loop

103. set\_pwm2\_duty(140); // Set PWM duty cycle

104. if(entrance) // If entrance sensor is activated

105. break; // Exit loop

106. }

107. while(exit); // Wait for car to fully exit

108. } else if (Start) { // If start button is pressed

109. if(Seconds != 0) { // If Seconds is not 0

110. while (Start); // Wait for start button release

111. break; // Exit GateControl loop

112. }

113. }

114. Delay10KTCYx(5); // Debounce start button

115. if(Seconds>4 || Seconds<0) // If Seconds is greater than 4 or less than 0

116. Seconds = 0; // Reset Seconds to 0

117. DispRomStr(Ln1Ch0,(ROM)"Enter parking lot"); // Display message on LCD

118. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

119. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

120. if(Seconds == 3 ) // If parking is full

121. DispRomStr(Ln1Ch0,(ROM)" parking is full"); // Display message on LCD

122. while(1){ // Inner loop

123. set\_pwm2\_duty(93); // Set PWM duty cycle

124. if(~exit || ~entrance) // If exit or entrance sensor is not activated

125. break; // Exit loop

126. }

127. motor = 0 ; // Set motor pin low

128. }

129. }

130.

131. void setup\_pwm(void) { // setup\_pwm function definition

132. // Set the PWM frequency and timer configurations for PIC18F45K22

133. // Assuming Timer2 with Prescale = 16 and Postscale = 1

134. PR2 = 155; // Set PWM period register

135. T2CONbits.T2CKPS = 0b11; // Prescaler 1:16

136. T2CONbits.TMR2ON = 1; // Timer2 ON

137.

138. // CCP1 and CCP2 as PWM

139. //CCP1CONbits.CCP1M = 0b1100; // PWM mode

140. CCP2CONbits.CCP2M = 0b1100; // PWM mode

141.

142. // Setting TRIS bits for CCP pins

143. //TRISCbits.TRISC2 = 0; // CCP1 output

144. TRISCbits.TRISC1 = 0; // CCP2 output

145. }

146.

**Proteus Simulation:**

The components used in proteus were the following:

* The microcontroller PIC18F45K22
* LCD
* Pull down resistors
* Motor
* LED
* Push Buttons
* Here, after pressing start, the motor is -2.16 which means the gate is closed, LCD shows that there is no cars yet in the parking lot and the LED is on

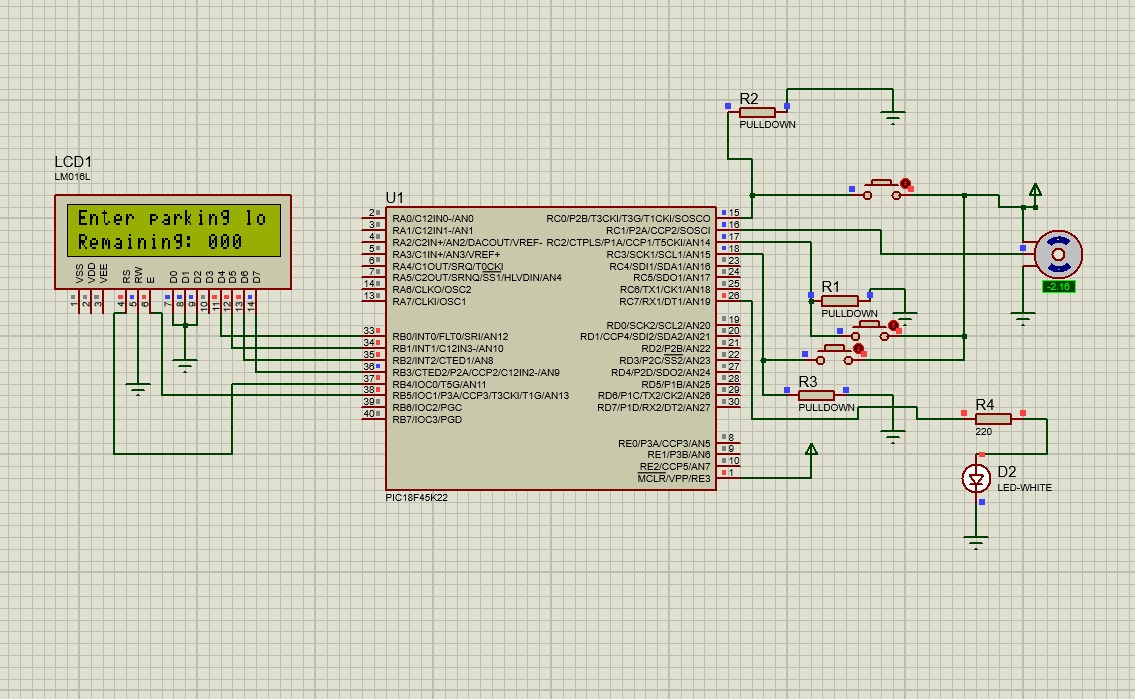


Figure 2: Initial state before pressing start

* After pressing the push button of entrance (which means in hardware that the first sensor before the gate detected a car), this results in opening the gate which is why the motor is 90

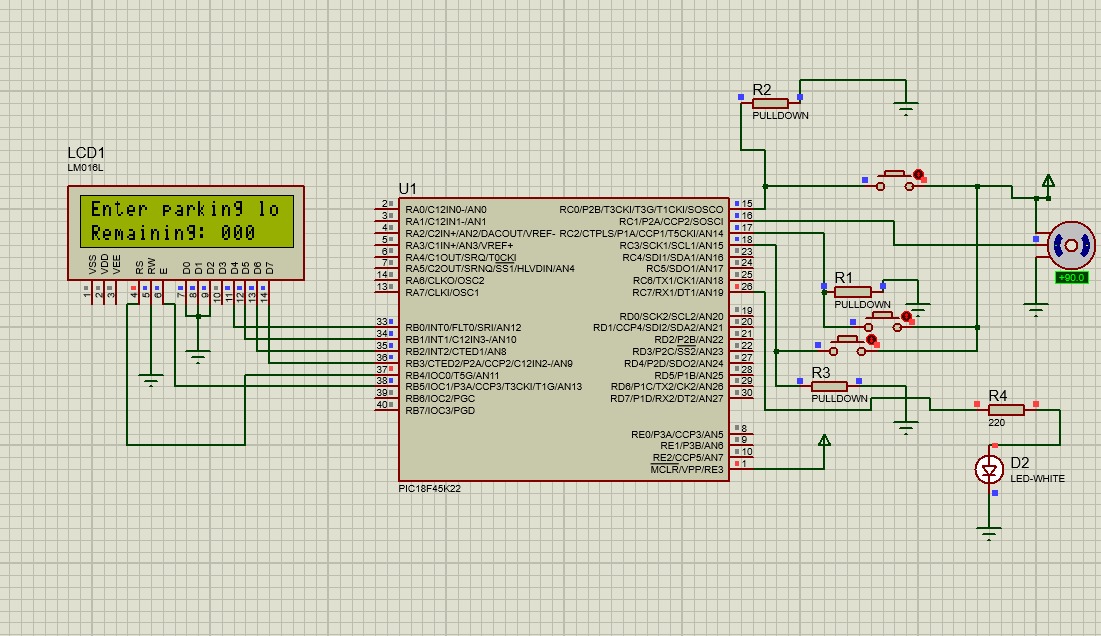


Figure 3: entrance push button is pressed

* After the push button of exit is pressed (which means in hardware that the second sensor which is after the gate detected the car so that means the car has entered the parking lot), the gate closes that’s why the motor is -2.16 and remaining becomes 001 which means there is one car in the parking lot so there are 2 more places left.

A computer circuit board with many wires

Description automatically generated

Figure 4: 1 car is in the parking lot

* The LCD display “Parking is full” so that the users can know that they can’t enter the parking lot (in hardware, first sensor detected 2 more cars so the car parking lot is full), the motor is -2.16 which indicates that the gate is closed at the moment. And if a user wants to enter the parking lot, he can’t because the gate won’t open

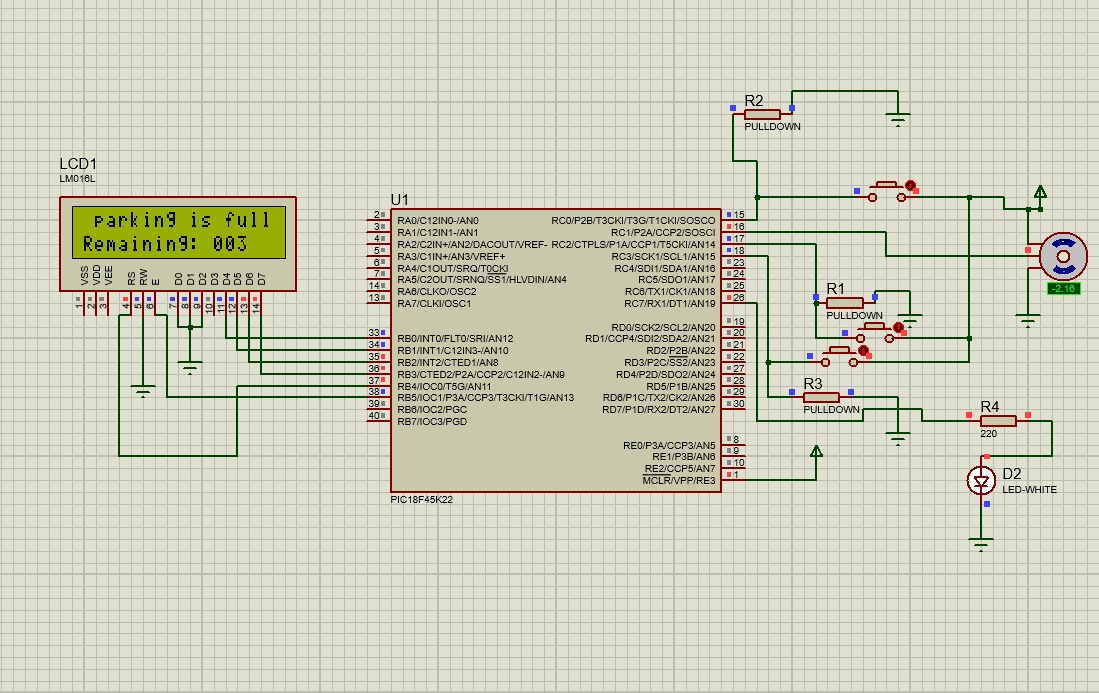


Figure 5: 3 cars are in the parking lot

* This means one car has left and 2 cars remain in the parking lot (in hardware, exit sensor detects a car, the gate opens, sensor entrance detects the car and the gate closes)

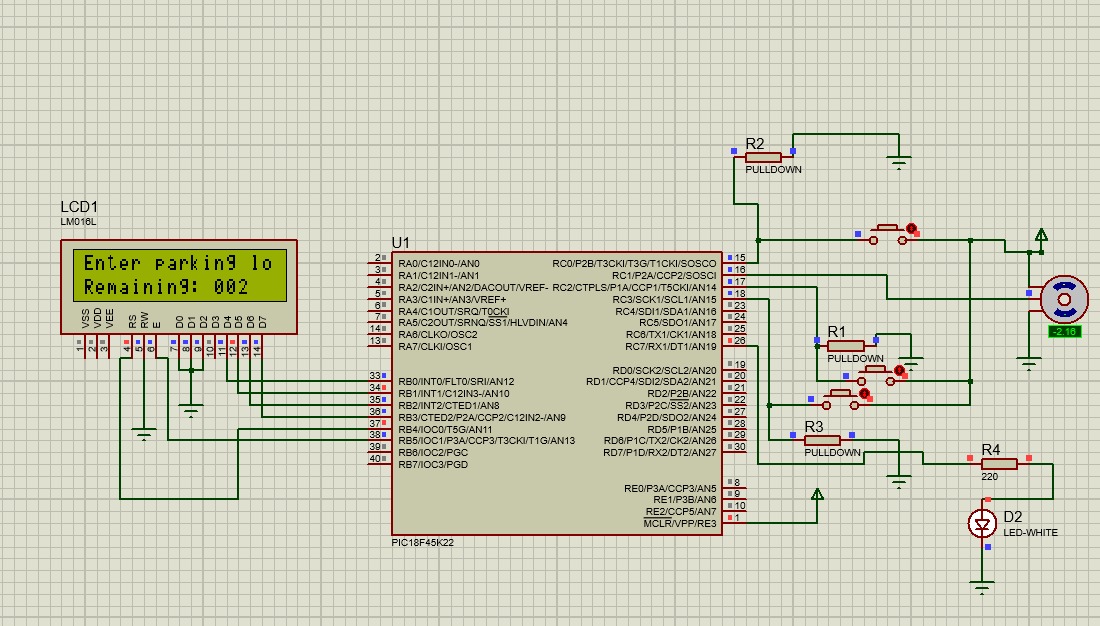


Figure 6: one of the 3 cars leaves the parking lot

**Hardware Code:**

1. #include <p18cxxx.h> // Include header file for the specific microcontroller being used

2. #include <LCD4lib.h> // Include library for interfacing with a 4-bit LCD

3. #include <EEPROM.h> // Include library for accessing EEPROM memory

4. #include <delays.h> // Include library for delay functions

5.

6. #define eeSeconds 0x00 // Define EEPROM address for storing Seconds variable

7. #define Device PORTCbits.RC7 // Define pin for controlling a device

8. #define Start PORTCbits.RC0 // Define pin for start button

9. #define exit PORTCbits.RC3 // Define pin for exit sensor

10. #define entrance PORTCbits.RC2 // Define pin for entrance sensor

11. #define motor PORTCbits.RC1 // Define pin for motor control

12.

13. #pragma config FOSC = INTIO67 // Configuration setting: Internal oscillator block

14. #pragma config LVP = OFF // Configuration setting: Single-Supply ICSP disabled

15. #pragma config BOREN = OFF // Configuration setting: Brown-out Reset disabled

16.

17. #define \_XTAL\_FREQ 8000000 // Oscillator frequency for delay functions

18.

19. #define plus PORTAbits.RA0 // Define pin for something named "plus"

20.

21. unsigned int ang1; // Declare variable ang1

22. int lmax, lmin; // Declare variables lmax and lmin

23. int i, servo = 1; // Declare variables i and servo, initialize servo to 1

24. char entered= 0, exited = 0; // Declare variables entered and exited, initialize to 0

25. char Seconds = 0; // Declare variable Seconds, initialize to 0 (represents number of cars)

26. char digits[3]; // Declare array to store digits

27.

28. void Setup(void); // Function prototype for Setup

29. void GateControl(void); // Function prototype for GateControl

30. void turnOn(void); // Function prototype for turnOn

31. void setup\_pwm(void); // Function prototype for setup\_pwm

32. void set\_pwm2\_duty(unsigned int duty); // Function prototype for set\_pwm2\_duty

33.

34. void main (void){ // Main function

35. lmin = 30; // Initialize lmin

36. lmax = 65; // Initialize lmax

37. ang1 = lmin; // Initialize ang1

38. while(1) { // Infinite loop

39. setup\_pwm(); // Setup PWM

40. Setup(); // Run setup function

41. turnOn(); // Run turnOn function

42. GateControl(); // Run GateControl function

43. }

44. }

45.

46. void Setup(void){ // Setup function definition

47. Seconds = 0; // Initialize Seconds to 0

48. TRISC |= 0x0D; // Set RC0, RC2, RC3 as input and RC1 as output

49. TRISCbits.RC7 = 0; // Set RC7 as output

50. ANSELC &= 0x70; // Configure RC0-RC4 as digital

51. InitLCD(); // Initialize LCD

52. DispRomStr(Ln1Ch0,(ROM)"Enter parking "); // Display message on LCD

53. DispRomStr(Ln2Ch0,(ROM)"Remaining cars are : "); // Display message on LCD

54. ReadEE(eeSeconds,&Seconds); // Read from EEPROM to get stored value of Seconds

55. Seconds =0; // Reset Seconds to 0

56. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

57. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

58. Device =0; // Set Device pin low

59. }

60.

61. void turnOn(void){ // turnOn function definition

62. while(1){ // Infinite loop

63. if (Start == 1) { // If start button is pressed

64. Device = 1; // Set Device pin high

65. Seconds = 0; // Reset Seconds to 0

66. set\_pwm2\_duty(93); // Set PWM duty cycle

67. return; // Exit function

68. }

69. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

70. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

71. }

72. }

73.

74. void GateControl(void){ // GateControl function definition

75. while(1){ // Infinite loop

76. while(Seconds == 3){ // Wait until all spots are full

77. if (~exit) { // If exit sensor is activated

78. Seconds--; // Decrement Seconds

79. while(1){ // Inner loop

80. set\_pwm2\_duty(140); // Set PWM duty cycle

81. if(~entrance) // If entrance sensor is activated

82. { Delay10KTCYx(20); // Delay

83. break; // Exit loop

84. }

85. }

86. while(exit); // Wait for car to fully exit

87. }

88. }

89. if (~entrance) { // If entrance sensor is activated (Car entering)

90. while(~entrance);// Wait for car to fully enter

91. while(1){

92. set\_pwm2\_duty(140); // Set PWM duty cycle

93. if(~exit){ // If exit sensor is activated

94. Delay10KTCYx(20); // Delay

95. break; // Exit loop

96. }

97. }

98. Seconds++; // Increment Seconds

99. while(~entrance); // Wait for car to fully enter

100. } else if (~exit) { // If exit sensor is activated (Car exiting)

101. Seconds--; // Decrement Seconds

102. while(1){ // Inner loop

103. set\_pwm2\_duty(140); // Set PWM duty cycle

104. if(~entrance) // If entrance sensor is activated

105. break; // Exit loop

106. }

107. while(~exit); // Wait for car to fully exit

108. } else if (Start) { // If start button is pressed

109. if(Seconds != 0) { // If there are cars in the parking lot

110. while (Start); // Wait for start button to be released

111. break; // Exit GateControl loop

112. }

113. }

114. Delay10KTCYx(5); // Debounce start button

115. if(Seconds>4 || Seconds<0) // If Seconds is greater than 4 or less than 0

116. Seconds = 0; // Reset Seconds to 0

117. DispRomStr(Ln1Ch0,(ROM)"Enter parking lot"); // Display message on LCD

118. Bin2Asc(Seconds,digits); // Convert Seconds to ASCII

119. DispVarStr(digits,Ln2Ch11,3); // Display Seconds on LCD

120. if(Seconds == 3 ) // If parking is full

121. DispRomStr(Ln1Ch0,(ROM)" parking is full"); // Display message on LCD

122. while(1){ // Inner loop

123. set\_pwm2\_duty(93); // Set PWM duty cycle

124. if(~exit || ~entrance) // If exit or entrance sensor is not activated

125. break; // Exit loop

126. }

127. motor = 0 ; // Set motor pin low

128. }

129. }

130.

131. void setup\_pwm(void) { // setup\_pwm function definition

132. // Set the PWM frequency and timer configurations for PIC18F45K22

133. // Assuming Timer2 with Prescale = 16 and Postscale = 1

134. PR2 = 155; // Set PWM period register

135. T2CONbits.T2CKPS = 0b11; // Prescaler 1:16

136. T2CONbits.TMR2ON = 1; // Timer2 ON

137.

138. // CCP1 and CCP2 as PWM

139. //CCP1CONbits.CCP1M = 0b1100; // PWM mode

140. CCP2CONbits.CCP2M = 0b1100; // PWM mode

141.

142. // Setting TRIS bits for CCP pins

143. //TRISCbits.TRISC2 = 0; // CCP1 output

144. TRISCbits.TRISC1 = 0; // CCP2 output

145. }

146.

147. void set\_pwm2\_duty(unsigned int duty) { // set\_pwm2\_duty function definition

148. CCPR2L = duty >> 2; // Set PWM duty cycle (8 MSBs)

149. CCP2CONbits.DC2B = duty & 0x03; // Set PWM duty cycle (2 LSBs)

150. }

151.

**Conclusion:**

In summary, the parking system effectively tackles the task of efficiently managing parking spots within a confined area, providing users with a smooth and convenient experience. Moreover, it served as a valuable opportunity for us to gain practical insights into working with the PIC18F45K22 microcontroller, illustrating the vast potential of microcontroller programming in diverse applications. The development and implementation of a smart parking system for a small parking lot with three parking spots present a significant advancement in modern urban infrastructure. By incorporating cutting-edge technology and innovative design, the system addresses the pressing challenges of managing parking spaces efficiently while enhancing the overall user experience.