EE447 TERM PROJECT

Muhammet Buğrahan KARA

2232148

ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

Laboratory Project:
Park Sensor System

Project Video Link: https://www.youtube.com/watch?v=_DbB8dWUzLo

1. INTRODUCTION

In this project, a car park sensor system which cooperates with driving safety components for a better driving experience is built. This system can also be connected to the car later in any time. The aim of this project is by using distance sensor, to measure the distance between car and obstacle and to brake if need. For this project, the tasks are sequentially:

1- A distance sensor HCSR-04 which is ultrasonic sensor located on the behind of the car. The sensors will continuously measure the distance to the obstacle (sidewalk or the front bumper of the car behind) and the system will take action based on the distance.

2- There is a threshold value which specify the maximum closeness to the obstacle from car. Then, according to this threshold value, car should stop if this threshold value are exceeded.

3- This system can be configured by the user and can be changed for his wish. For example, system threshold value can be adjusted and user remove braking condition and continue driving. Also, these information such as threshold, distance to threshold are shown on LCD screen.

4- To realize this system, a stepper motor will be used and to adjust threshold, a potentiometer will be used. Stepper motor is turning fast inverse proportionally to the distance. If the distance is close to threshold, motor is turning slow; else is turning fast.

5- Also, the threshold can be decided by using a “keypad” whose 10 buttons can be assigned to digits from 0 to 9. Instead of adjusting potentiometer value as threshold, user can directly give the threshold value to system by pressing right buttons of

keypad.

6- By using **TM4C123G Board,** all the system is brought together. The buttons on this board is used to control operation modes.

1. PROJECT SOLUTION

This project is implemented in three phases for simple and well disciplined way. For the first phase of this project, except LCD screen, other parts of this project are handled like driving motor, measuring distance. For the second phase of this project is to make LCD screen to work and show the necessary information. The third one is to bring all parts together and provide an efficient work with all. Also, some explanation about BONUS part, will be in third phase.

Note: To be more understandable, project will be explained item by item generally and sequentially.

Note: Keypad Bonus Part and Changing Speed According to Distance Bonus Part is done. I will explain them in future explanations in phase3.

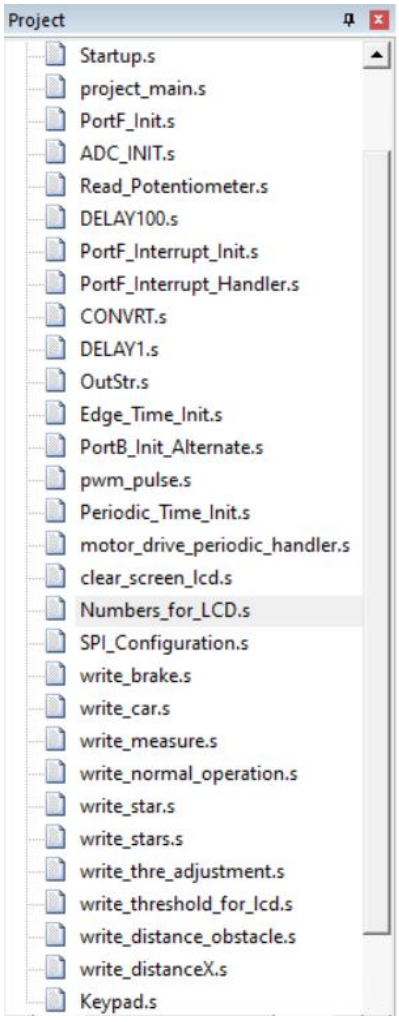

Figure 1: List of Files of Project

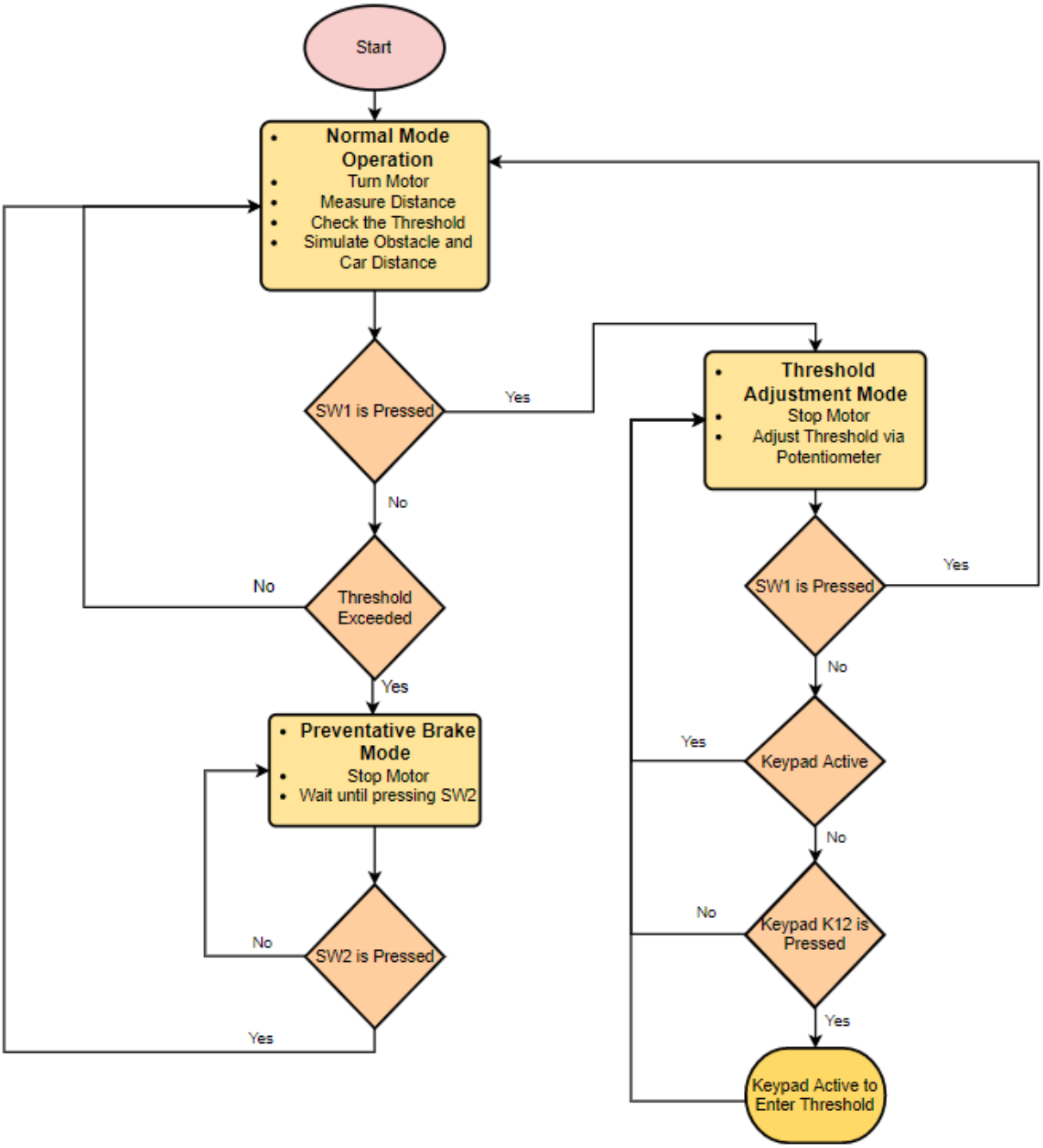

Figure 2. Flow Chart of Project

2.1.Phase1

For this part of the project, it can be said that the setup except LCD is configured and set. This setup includes these parts separately:

- "Read measurement from distance sensor"
  - To read distance to obstacle, HCSR-04 is used as ultrasonic sensor which has 4 pins: Vcc, Trig, Echo, Gnd. Vcc are connected to the VBus pin of the Tiva Board like Gnd pin which is connected to Ground pin of board.
  - Trig pin of sensor is connected to F2 pin of board. Trig pin is used to trigger the sensor when we want to read a measurement, so a pulse is created on Board to be used in triggering function. This pulse is created by using PWM function GPTM Timer1A. This PWM is initialized in "pwm_pulse.s" file and the PWM is obtained from F2 pin. This pin is also internally connected to Blue Led, so pulse can be observed with embedded led on the board. Mainly, this pulse is triggering the sensor and make it send a new wave to measure the distance later.
  - Echo pin is used to detect the wave which returns after hitting an obstacle. To detect them, a timer Timer0A of GPTM works in edge time mode and B6 pin of board is used in capture edge time mode between alternate functions.Timer0 RIS register is controlled in the main of project and TAR values are stored to calculate the time between two consecutive detected edges.
  - Finally, the distance is calculated in every loop in main file in this project. Also, 1s delay was put between two consecutive calculations to obtain well results.
    - ✓ pwm_pulse.s
    - ✓ main.s
    - ✓ Edge_Time_Init.s
    - ✓ PortB_Init_Alternate.s

- "To use SW1 and SW2 in interrupt, Open Interrupt GPIO of F port"
  - SW1 and SW2 exist on the board already, so they can be fully functional if they are configured according to wish. One of the aim of this project is to adjust threshold value by pressing SW1 and by pressing SW2 to remove braking condition. As the switches can be pushed anytime, following them via interrupt is more sensible. Then for F port, GPIO interrupt is initialized in "PortF_Interrupt_Init.s". As F0 and F4 pins are connected by pull up resistors, interrupt occurs in falling edge.
  - For this interrupt, priority is given as 2, so it has larger priority comparing to motor_drive_periodic_handler.
  - When two switches are pushed, some functions should be realized like "read potentiometer values" or "removing braking condition", so these functions are performed in interrupt handler which will be active when one of the key is pressed. These functions will be explained in future subparts.
    - ✓ PortF_Init.s
    - ✓ PortF_Interrupt_Init.s
    - ✓ PortF_Interrupt_Handler.s

- "Read Potentiometer as Threshold Value"

- To adjust threshold, a potentiometer is used. Potentiometer result is analog, so result should be converted to digital. To aim this, ADC property of board is used, so E3 pin of board is configured as analog input and ADC is set on board in file “ADC_Init”.
- For software triggering as trigger source, analog read should be started when reading is intended. Using “PSSI” register, analog measurement is started, and board convert analog value to the digital value. Then the converted result can be found in “SSFIFO” register. This value can be assigned as threshold value.
- After pressing SW1 first time, this “Read_Potentiometer.s” subroutine will become active and until second press to SW1, it continues to read new threshold value from potentiometer.
    - ✓ ADC_Init.s
    - ✓ Read_Potentiometer.s

- “Drive Motor”
    - To drive motor, a GPTM periodic timer Timer3A is created and its handler is used to turn motor for each step of motor when every periodic time is up.
    - For this interrupt, priority is given as 3, so it has smaller priority comparing to PortF Interrupt.
    - As stepper motor is used, periodic timer is used to create delay for time between each step. As requirement 5ms, in up counter mode, a prescale is used because prescale behaves timer extension in up count mode, so timer becomes 24 bit. With 16 bits, we can not reach 5ms, and we can reach at most 4ms.
    - (BONUS Related) In this project, Timer has at least 0x01F000 as reload value, so it is nearly 7.8ms. According to distance, reload value can be increased until to 0x0FF000(16+8bit,TAPR+TAILR) by incrementing prescaler interval register “TAPR”. When other interval values are used, speed change may not be observable sometimes, so for the best interval is accessible by changing TAPR between 0x0F-0x01 values.
    - To stop motor, timer is disabled and give 0 to output pins which drive motor.
    - To drive motor, PortB pins B0-B1-B2-B3 are used.
        - ✓ Periodic_Time_Init.s
        - ✓ motor_drive_periodic_handler.s
        - ✓ PortB_Init_Alternate.s

2.2.Phase2

For this part of the project, LCD is configured and set. Results and information are written on LCD. Before writing onto LCD, talking about SPI and LCD will be a better choice.

To use LCD, SPI must be known because the communication protocol of this LCD is SPI. The SSI is used to send synchronous serial communication to other devices, and can be configured to follow various protocols. SSI means Synchronous Serial Interface and there is master-slave relation. In this case, Tiva board is master and LCD screen is slave. Master sends data and slaves receives data. When sending, the data is sent loaded into a FIFO buffer, and sent out according to the configured bit rate. This communication needs 4 wire such as MOSI, CLK, Fss, MISO but in this case, we don’t need to MISO wire because board is not receiving

anything from LCD. Due to synchronous communication system as common clock, the CLK wire should be connected. Fss pins shows the slave selection, it is given 0 to select slave.

For the Nokia 5110, there are 8 pins which are connected to the board like that:

1. VCC Pin ➔3.3V
2. GND Pin ➔ Ground
3. SCE Pin CHIP ENABLE to control operation of Pin Controllers ➔ Fss pin=A3
4. RESET Signal RESET for operation of LCD ➔A7
5. D/C Pin to configure the data formats between Data and Command. ➔A6
6. SDIN Pin DATA (SERIAL DATA LINE)➔Tx pin=A5
7. SCLK Pin CLOCK (SERIAL CLOCK LINE) ➔Clk pin=A2
8. LED Pin to control operation of LED (Back Light) ➔---

The address arrangement of memory that is shown on LCD Display (DDRAM) is Matrix that consists of 6 rows (Y Address) from Y-Address 0 to Y-Address 5 and 84 columns (X Address) from X-Address 0 to X-Address 83. We can write data into the address of memory (DDRAM) continuously and values of X-Address and Y-Address will be increased automatically.

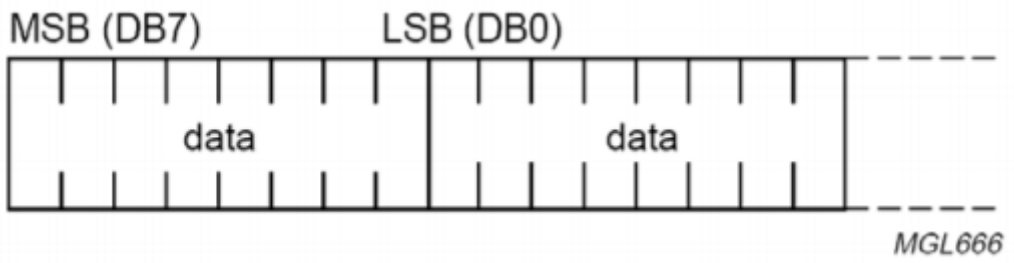

Figure 3. Sending Datas to LCD bit by bit

The format of command that is used to communicate with LCD is divided into 2 modes; Command Mode and Data Mode. In this case, it uses Pin D/C to divide and control signals; if D/C = 0, the data that is sent to LCD is Command and if D/C = 1, the data that is sent to LCD will be Data and it will placed in DDRAM Memory (Display Data RAM) to be displayed on LCD Display. After 1 byte data has already been written, 1 value of DDRAM address will be increased automatically. The format of data will be serial and it will send MSB (The Most Significant Bit) first. Pin SCE is still Status LOW (SCE = 0) until data byte will be sent successfully.

- "SPI Function of Board is Active"
  - To communicate between LCD and board, firstly the SPI alternate function of board should be opened. So, firstly the GPIO Port A should be configured to use as SPI.
  - To start SPI, we should make configuration SPI like that:
    - Choose 2Mbit/s:

SSInClk = SysClk / (CPSDVSR * (1 + SCR))
2x10^6 = 16x10^6 / (CPSDVSR * (1 + SCR))
2x10^6 = 16x10^6 / (2* (3 + 1))
Note that : LCD max:4Mbit/s

- SSI0_CC➔Clock source is chosen PIOSC=16Mhz with 0x5
- SSI0_CPSR➔CPSDVR=2, it should be even number.
- SSI0_CR0➔SCR=1,SPH=0,SPO=0,Freescale Format,8 bit data
- SSI0_CR1➔SSI Enable
- CE_Fss_A3_pin➔Give 0 to select slave

✓ SPI_Configuration.s

- "LCD Configuration to Make it Active"
  - To work LCD, firstly some commands are sent to the LCD.
  - To start SPI, we should make configuration SPI but before it, we firstly choose DC_A6_pin as 1 to say that " coming data is command" and we check SSI0_SR status register whether the communication is BUSY or not. If not, we can send our data as byte.
  - To send command, we store the byte which will be sent in SSI0_DR.
    - Chip active,horizontal addressing mode (V = 0); use extended instruction set (H = 1): 0x21
    - LCD Vopset contrast configuration: 0xB1
    - Temperature control value setting: 0x04
    - LCD Bias Mode setting: 0x13
    - Basic Command Mode/Basic Display Control Mode: 0x20
    - Set display control to normal mode: 0x0C
    - Set cursor X:0 =0x80
    - Set cursor Y:0=0x40

✓ SPI_Configuration.s

To write data onto LCD to show on the screen, the data as byte is sent to LCD via SSI0_DR register. When the data is written onto LCD, screen part where the cursor points becomes dark according to given data. In one column at the row, there are 8 pixels, so according to received data, it makes some pixels dark.

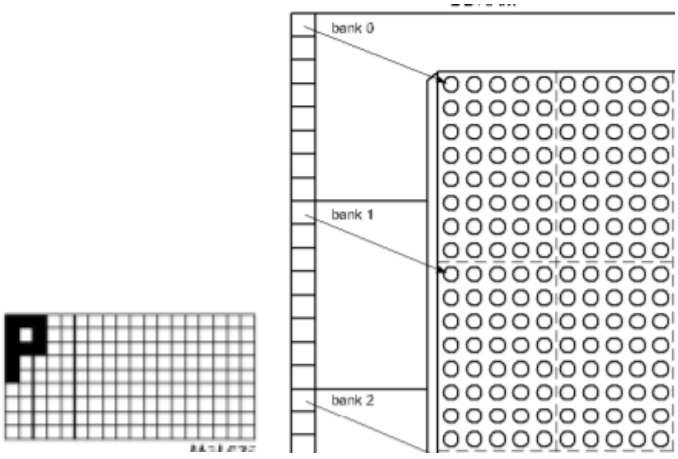

Figure 4. LCD screen with pixels

Characters and numbers are shown with 5 bytes which is specified before, so in this project, these bytes are stored in memory firstly, then they are sent to LCD screen via SPI. But before each sending data, Status Register is controlled whether TNF(Transmit Fifo Not Full) is 1. If it is 1, then new data can be loaded to DR.

- "Showing Data On LCD"
  - To write the intended character or numbers on LCD screen, firstly specified 5 bytes for each consecutive characters are saved to specified location in memory.
  - Then the data in this location memory are storing into Data Register sequentially but waiting enough time by checking Status Register TNF flag.
  - If the words should be written to second or third row, first thing to apply is sending command to LCD to change cursor location according to figure4 given below.

INSTRUCTION	D/$\overline{C}$	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	writes data to display RAM
(H = 0)										
Reserved	0	0	0	0	0	0	1	X	X	do not use
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Reserved	0	0	0	0	1	X	X	X	X	do not use
Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	do not use
	0	0	0	0	0	0	0	1	X	do not use
Temperature control	0	0	0	0	0	0	1	TC_1	TC_0	set Temperature Coefficient (TC_x)
Reserved	0	0	0	0	0	1	X	X	X	do not use
Bias system	0	0	0	0	1	0	BS_2	BS_1	BS_0	set Bias System (BS_x)
Reserved	0	0	1	X	X	X	X	X	X	do not use
Set V_{OP}	0	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	write V_{OP} to register

Figure5. LCD Command/Data Table

  - To write some words and numbers, subroutines are written to use them in more easy way. Then these subroutines are listed like that:
    - ✓ write_brake.s➔ write to LCD as "Brake On"
    - ✓ write_car.s➔ write to LCD as "Car"
    - ✓ write_measure.s➔ write to LCD as "Meas(mm):"
    - ✓ write_normal_operation.s➔ write to LCD as "Normal Op"
    - ✓ write_star.s➔ write to LCD as "***"
    - ✓ write_stars.s➔ write to LCD as "**********"
    - ✓ write_thre_adjustment.s➔ write to LCD as "Thre. Adj."
    - ✓ write_threshold_for_lcd.s➔ write to LCD as " Thre(mm):"

  - Above ones are constant but below ones are shows the simulation of distance and threshold on the screen, so they can be changed. For them, firstly, all columns of row is filled by "-", and according to threshold location, cursor

location is changed and write “X” there. Then according to obstacle distance, “|” will be writen onto other characters by updating the location of cursor according to obstacle for each control loop.

- ✓ write_distanceX.s➔ write distance onto LCD as "--X------"
- ✓ write_distance_for_lcd.s➔write distance onto LCD as "||||||||"

2.3.Phase3

For this part of the project, all systems is brought together and details for BONUS part is explained.

- ❖ “BONUS Parts”
  - ○ “Changing Speed According to Distance”

➢ Speed should be changed according to distance and if the distance to obstacle is too high, then motor should be turn fast. Else motor should be slow down until threshold. To change the speed of motor according to distance as mentioned before, Prescaler Register Timer3 is used to increase period or decrease period. In up count mode, prescaler behaves like time extension, so if prescaler interval value is increased, then the waiting time for each step of motor will be increased. According to this logic, below algorithm is used:

```
SUB         R1,R8,R7 ;R1: difference of threshold and
distance ;an algorithm to speed up or down

MOV         R0,#0x0E ;ADD MAX 0x0E+0x01 to TAPR

MUL         R1,R0
MOV         R2,#1000 ;1000 is the max  difference and
            equal to 0x0FFF
UDIV        R1,R2
SUB         R1,R0,R1 ;R1:Will be added part to Prescaler
            Interval Register
LDR         R0,=TIMER3_TAPR
MOV         R3,#0x01
ADD         R1,R3
STR         R1,[R0] ;load reload value for prescale
            register
```

➢ In this algorithm, as mentioned before, timer3 interval has at least 0x01F000 as reload value, so it is nearly 7.8ms. According to distance, reload value can be increased until to 0x0FF000(16+8bit,TAPR+TAILR) by incrementing prescaler interval register “TAPR”. When other interval values are used, speed change may not be observable sometimes, so for the best interval is accessible by changing TAPR between 0x0F-0x01 values. So, firstly difference between threshold and distance is calculated and max difference “1000” is reversely proportional to adding max values 0x0E to TAPR. For example difference may 400mm, then adding value to TAPR will be 0x0E-(400*0x0E/1000) to slow motor down.

- ✓ motor_drive_periodic_handler.s

  - ○ “Changing Threshold By Using Keypad”

➢ The user could set the threshold distance by entering three decimal digits using the 4x4 keypad. In this case, after pressing SW1, user can change threshold value by using potentiometer but if the user should press a button “K12” on the

keypad for 1 sec to indicate that the threshold will be set using the keypad. K12 value is pooled when threshold value can be changed by potentioemeter. When the user pressed K12, then now threshold value is taken from keypad.

- The logic behind of Keypad is:
    - Giving rows as outputs as 0 initially.
    - Columns are defined as inputs and have pull up resistors, so when the key is pressed, input will be ground 0.
    - Making rows sequentially output 0 by one by while other is 1, helps to find which key is pressed.
    - Then after finding the pressed key, first pressed one multiplies with 100. Then the second key is multiplied by 10 and added to previous value. The last key is added to previous value.
    - User can enter threshold value by pressing 3 times every time but user can also be enter 2 or 1 digits. It means that when the user pressed one key firstly like 5, threshold will become 500 if any other key is not pressed and SW2 is pressed.
    - For example, firstly press 4, then secondly 2. ➔Threshold=420
    - Firstly press 2,secondly 3,thirdly 5➔Threshold=235
    - Firstly press 1➔Threshold=100
    - Until pressing SW2, it continues like that.
    - Only {K1-K12} buttons are used. K1=0,K2=1…K9=8,K10=9. When the user press K11 and K12 while threshold mode, the keys does not enter any value, pass this press.

- "System General Process"

  For all systems, they can be summed like that:

  - In main file, the distance is always measured by pooling Timer0 capture edge RIS. Then the distance is found in main loop every time. Finding the distance, digits of the distance is found by one by and it is send to "Numbers_for_LCD.s" subroutine to write on the screen with necessary words on LCD like "Meas(mm):". To write necessary words, also some writing subroutines like write_measure are called to write on the screen what we want.
  - Initially threshold is defined by 100, so after pressing SW1, the system goes to interrupt handler of portF. Firstly interrupt handler is finding the key which is pressed, then if SW1 was pressed, "Read_Potentiometer" subroutine is called. Also to follow SW1 push times, a flag is incremented, if this flag=2, then system understand that it should return normal mode. When adjusting threshold, motor turns off. In "Read_Potentiometer" subroutine, by starting PSSI, an analog read is made. Then this result is converted a result between 0-999. Again, the Numbers_for_LCD is called after finding the digits of result by one by and it is written on LCD. After pressing SW1 second times, the last result is stored in R7 register as new threshold value.
  - When the new threshold is adjusted with threshold in Thre. Adj. Mode, if the user pressed K12 button of Keypad, then new threshold value is given by keypad. User can enter one of K1-K10 buttons and it will be seen on the screen according to rules mentioned above in Bonus part.
  - If the distance is changing, the speed of the motor is also changing according to distance as mentioned in Bonus part.
  - System always check the difference of threshold(R7) and distance(R8) before

driving motor. If the threshold is exceeded, then system disable Timer3 which is used to drive motor and also give motor inputs 0 and a flag is used R10 to indicate the system is in braking mode and call “write_brake” subroutine to write on LCD “Brake ON”.

- If the SW2 is pressed, in Interrupt handler of portF, timer3 is again started and enabled to turn motor again and return Normal Mode.
- If the motor is turning normal, then the system call “write_normal_operation” to show it on LCD and by calling other words “write_...” everything is seen on LCD screen.
- For priority of interrupt, system has the highest priority for port F and the second high priority level is motor driving handler.
- General working principle is like above.
- The connections are given below in table.

Motor Driving	B Port
IN1	B0
IN2	B1
IN3	B2
IN4	B3
Periodic Timer	Timer3A
Keypad Threshold Setting	D port + E port
R1	E0
R2	E1
R3	E2
R4	E4
L1	D2
L2	D3
L3	D6
L4	-
LCD	A Port
RST	A7
CE	A3
DC	A6
DIN	A5
CLK	A2
Vcc	3.3V
BL	-
GND	GND
Distance Sensor	F port+B port
TRIG	F2+Timer1A PWM Mode
ECHO	B6+Timer0A Edge Capture Mode
Potentiometer ADC	E port
Analog Input	E3

Figure6. Normal Operation

Figure7. Threshold Adjustment Operation

Figure8. Brake ON Operation

2. CONCLUSION

This project is implemented in three phases for simple and well disciplined way. First phase is about the system function like turning motor, measuring distance, while the second phase is about writing information onto LCD screen by using SPI protocol. Third phase is about all system and Bonus parts.

In this project, a park sensor system is implemented. By using ultrasonic sensor, the distance are measured and compared with adjustable threshold value. According to difference with them, system can slow motor down or speed up. If the threshold is exceeded, system stop motor, so hitting obstacle cases are prevented. This system can be embedded to already existed cars and it will be work fine with car information screen in front of driver. This system also simulate the distance of obstacle and threshold on the screen, so it will easies the parking of car. Setup of the system is shown in figure 6.

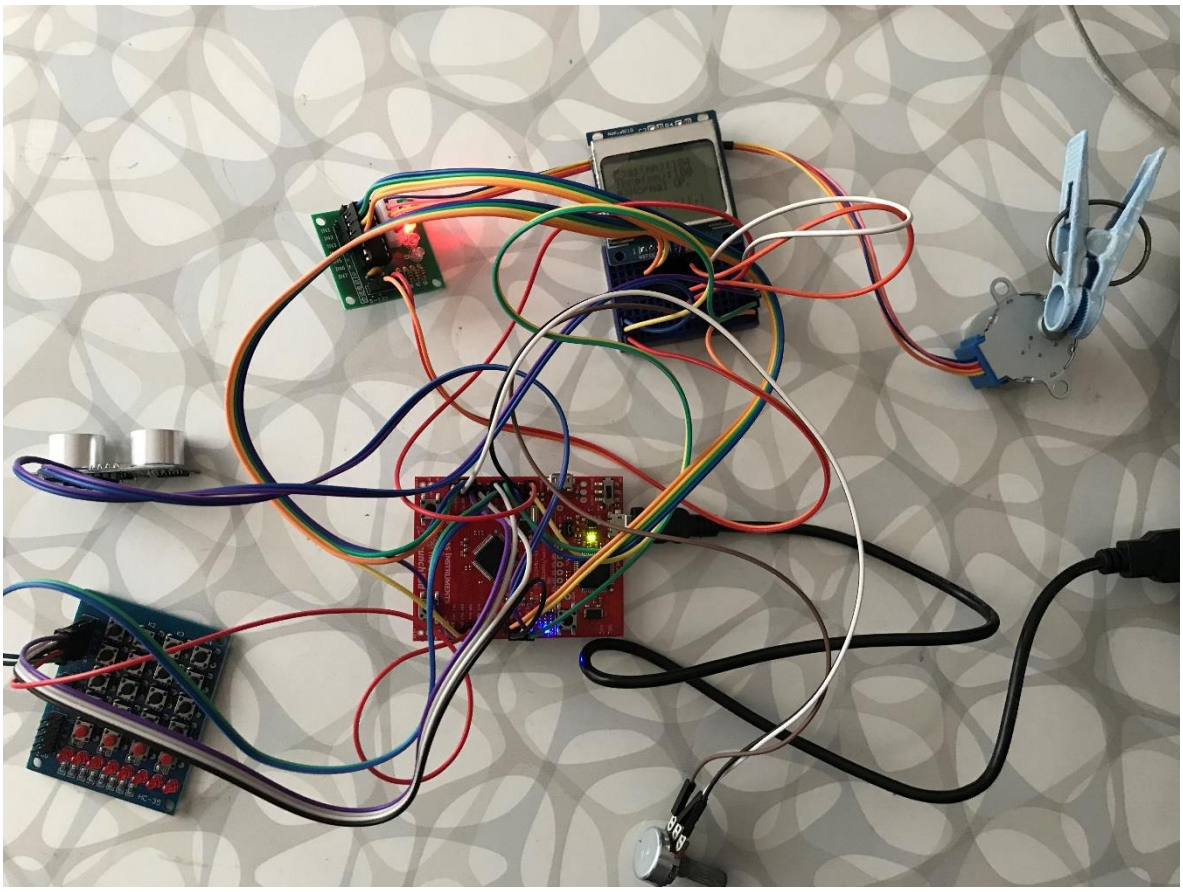

Figure 9. Setup of Project

Project Video Link: https://www.youtube.com/watch?v=_DbB8dWUzLo