ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

Laboratory Project: Park Sensor System

Laboratory Project

Objectives

In the EE447 laboratory work, you were expected to familiarize yourself with the operation of TM4C123G and its utility modules. Now, in this final project you are expected to gather the previous experience on the microcontroller with novel information to achieve a multi-functional task. The objectives of the project are as follows:

- Interpretation of the necessities of complex task and encapsulation into sub-task
- Fulfillment of co-operation of utility modules
- Understanding a given complex hardware and compatibility of its components
- Writing a multi-task software for a given complex set up
- Introduction of the serial communication on TM4C123G and utilization of the facility on SPI protocol.

1 Project Definition

In this project you are expected to build a car park sensor system that cooperates with driving safety components for a better driving experience. This useful system is found in almost all modern cars with varying degrees of functionality. In some cases it is sold as a separate upgrade module to bring park sensor functionality to cars that do not have it from factory.

The system has three major functions:

1.1 Ultrasonic Sensing

There can be multiple ultrasonic sensors located on the periphery to provide information from all sides 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. If it is closer than a user specified threshold, the car will brake and the user will be visually informed about the situation.

1.2 Preventative Braking

The car needs to stop if an obstacle is too close. The brakes can be engaged immediately when the threshold is exceeded, or they can be engaged proportionally as the distance gets smaller, provided that the car fully stops when the limit is exceeded.

1.3 User Interface

The user needs to set the threshold of what is considered a safe distance from the obstacles. The user should also be able to remove the braking condition at will and continue driving the car. An information screen will show the user the threshold distance, the currently located obstacle(s), and the state of preventative brakes (whether engaged or not).

2 Requirements and Restrictions

The overall functionalities of park sensor system are described in 1. For the sake of simplicity there will be some assumptions about operation. Additionally, there will be restrictions in both the implementation and the hardware to be used.

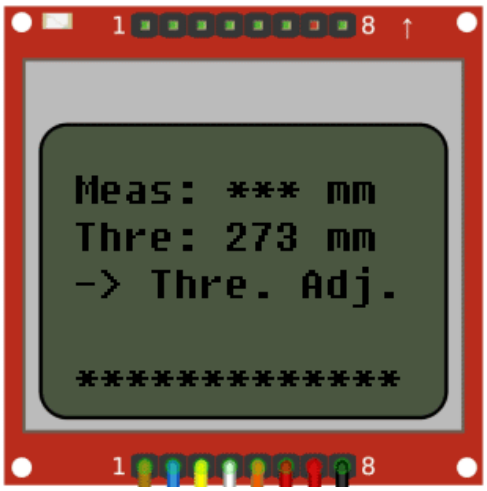
2.1 Requirements

You will use the Nokia 5110 LCD screen as the main user interface, HC-SR04 as the distance sensor, a potentiometer for threshold setting, and the stepper motor as a symbolization of the car movement. Following are the functional requirements:

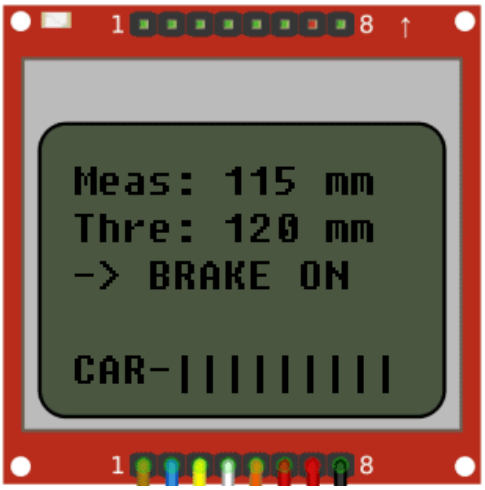
1. The user should be able to see the currently configured distance threshold on the screen in millimeters with 3 decimal digits.
2. The user should be able to see the currently measured distance on the screen in millimeters with 3 decimal digits.
3. Distances are to be shown in millimeter units with three decimal digits. e.g.: `111 mm`. The maximum distance should be limited to 999 mm.
4. The user should be able to see the currently measured distance on the screen as a progress bar at the bottom of the screen which visually indicates the current distance and the threshold to scale between 0 mm and 999 mm. See below for *an example progress bar idea.*
5. During normal operation mode, the threshold will stay constant and only the measurements will be updated. The threshold will only be changed in threshold setting mode. A possible UI design for normal operation mode is given in the Fig. 1a.
6. The user will press the onboard button SW1 to enter threshold setting mode. In this mode, the thresholds proposed value will be read from the potentiometer and viewed on the screen frequently while the user is adjusting. A second press to SW1 will set the threshold and exit threshold setting mode. A possible UI design for threshold setting mode is given in the Fig. 1b
7. **(BONUS)** Optionally, the user could set the threshold distance by entering three decimal digits using the 4x4 keypad. However, this does not replace the original method of setting the threshold using the potentiometer. In this case, after pressing SW1, the user should press a button (any button or a button you choose, up to you) on the keypad to indicate that the threshold will be set using the keypad. A second press to SW1 will set the threshold and exit the threshold setting mode.
8. During normal operation, when the threshold is exceeded, the system will enter the preventative braking mode. A possible UI design for preventative braking mode is given in the Fig. 1c.
9. The user should be visually informed when the preventative brakes are engaged. The visual indication could be text based or any other reasonably sized visual indication that can be clearly read or seen by the user. For example, when the brakes are engaged, the display could say `"BRAKE ON"` or it could be filled with completely dark pixels.
10. The user should be able to remove (reset) the preventative braking state by pressing the onboard button SW2.
11. The stepper motor should continuously rotate in normal operation mode, that is, as long as the threshold is not exceeded after a reset. The motor should be stopped if the preventative brakes are engaged and stay still until the system is reset (by pressing SW2, not to be confused with the global reset button).
12. **(BONUS)** Optionally, the step motor speed could be adjusted according to the distance to obstacle. The motor should slow down as the obstacle gets closer to the threshold, and come to a full stop when the threshold is reached as dictated by the previous requirement.

An example progress bar idea: A text-based progress bar may look like "`--X---||||`" where each character space represents a 100 mm distance. `X` indicates the threshold distance which is between 200 and 299 since it is the third space from the left. The pipes `|` on the right indicate the approximate position of the obstacle. Here, since every space represents 100 mm, the obstacle is between 600 and 699. As the obstacle gets closer, the pipes `|` will approach the threshold `X`. For example, if the threshold is set to 120 mm, and the obstacle is at 310 mm, then the progress bar would look like "`-X-|||||||`". **This is just an example progress bar, you can stylize it to your liking as long as it visually indicates the threshold and the measured distance in proportion between 0 mm and 999 mm**.

(a) Normal mode UI



(b) Threshold setting (adjustment) UI



(c) Preventative braking mode UI

Figure 1: Park Sensor system operation mode user interface designs

2.2 Restrictions

2.2.1 Hardware Restrictions

The park sensor system is uses the following components:

1. NOKIA 5110 LCD Screen
2. 1 Potentiometer
3. 1 4x4 Keypad (optional)
4. 2 Buttons Placed on the TM4C123G Board
5. HC-SR04 Ultrasonic Sensor
6. Stepper Motor

2.2.2 Implementation Restrictions

To ensure the learning outcomes of the course and this laboratory, there are a few restrictions on implementation:

1. Potentiometer must be read using the ADC module.
2. Stepper motor must be driven using GPTM interrupts. That is, the motor signals should advance one step in the determined direction (left to you) every time in a GPTM interrupt handler that you will configure. For reasonable operation of the stepper motor, you should advance steps no sooner than 5 milliseconds. This way you can stop the motor by disabling the timer, and adjust the speed by changing the reload register value.
3. GPIO interrupt flags must be used to detect button press events of onboard buttons SW1 and SW2. You are free in handling the 4x4 keypad optional feature. Note that you must use the onboard buttons SW1 and SW2, you are not allowed to use the keypad for threshold setting and reset operation buttons.
4. HC-SR04 ultrasonic sensor must be read using a GPTM in input edge time mode.

The visual style of the user interface on Nokia LCD screen is up to you as long as you fulfill the requirements specified in Section 2.1. To be eligible for bonus points from the project, you have to fulfill the requirements described in Section 2.1 by satisfying the restrictions described in Section 2.2.

3 Tips

On board buttons are connected to pins PF0 and PF4 [1]. Note that PF0 is locked [2] and you should unlock that pin to program it. You might want to refer pages 684 and 685 of the TM4C123GH6PM manual [2].

You will need to use interrupts and configure the priority of the interrupts. Recall that to configure an interrupt, you should know the interrupt number of the interrupt source you plan to use so that you can decide which NVIC registers (see page 141 of [2]) to be configured. You can find the interrupt number of an interrupt source from the table in page 104 of [2].

Under the given restrictions, you will drive the stepper motor using a GPTM interrupt handler. Thus, you may do other repetitive tasks in an endless loop inside main. An example algorithm for the main loop is given below.

1. Check the system state (normal operation, threshold setting or preventative braking)
2. If preventative breaking, then wait for a reset. If resetted, then return to normal operation
3. If threshold setting, read the user adjustment (potentiometer), update the value on screen, if threshold set button is pressed then return to normal operation
4. If normal operation, then measure the ultrasonic sensor distance, update the value on screen
    - If threshold distance is exceeded, then stop the motor and enter preventative braking state
    - If threshold set button is pressed, then enter threshold setting state
5. Loop back to 1.

4 Background Information: Serial Peripheral Interface

Serial transmission involves sending one bit at a time, such that the data is spread out over time. Compared to parallel communication, many fewer lines are required to transmit data. This requires fewer pins but adds complexity. Serialized data is not generally sent at a uniform rate through a channel. Instead, there is usually a burst of regularly spaced binary data bits followed by a pause, after which the data flow resumes. Packets of binary data are sent in this manner, possibly with variable-length pauses between packets, until the message has been fully transmitted.

In synchronous systems, separate channels are used to transmit data and timing information. The timing channel transmits clock pulses to the receiver. Upon receipt of a clock pulse, the receiver reads the data channel and latches the bit value found on the channel at that moment.

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems and in this project it will be used. SPI involves a master and a slave, or multiple slaves. SPI interfacing involves 3 or more wires, consisting of a clock, serial data out, serial data in, and chip select if necessary. The master MCU basically sets the clock rate for the slaves, asks a specific one to listen up using the chip select port, and sends them commands via its serial data out port, and expects to receive the output from the slave through the serial data in port.

4.1 Synchronous Serial Interface in the TM4C

The TM4C has four Synchronous Serial Interface modules (SSI). The SSI is used to send synchronous serial communication to other devices, and can be configured to follow various protocols. We will use SPI in this lab, which requires that we specify which device will be sending data (Master), and which device(s) will be receiving data (Slave). When sending, the data is sent loaded into a FIFO buffer, and sent out according to the configured bit rate. Each FIFO buffer is 16 bits wide, and 8 locations deep. The size of the data can be configured to be from 4 to 16 bits wide depending on your needs.

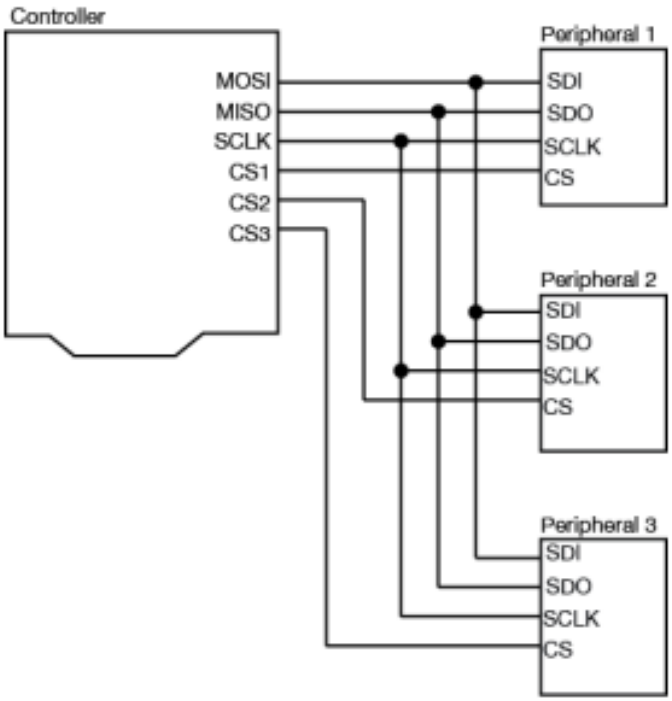


Figure 2: Synchronous Serial Interface in the TM4C

The pin connections for all SSI ports are labeled in Figure 2. When using Synchronous Serial Commu-

nication, there are typically 4 pins (lines).

- RX (MOSI) – Receiving data line
- TX (MISO) – Sending (transmitting) data
- Clk (SCLK) – The clock each bit is synched with
- Fss – Used to tell the slave that data is being sent

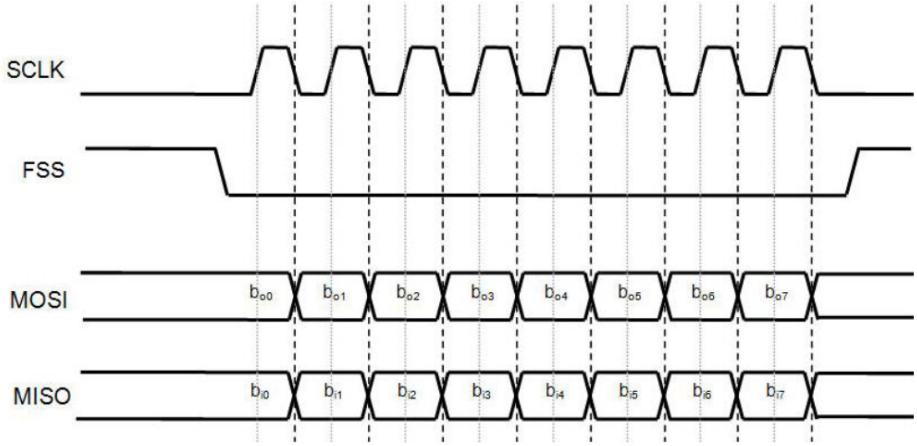


Figure 3: How SPI signals change as data is sent

4.2 SPI Configuration

In order to have Nokia 5110 functioning, 3 separate configurations are required, first of which is GPIO where the corresponding I/O pins are initialized to work as SPI pins. In the second part, the SPI module on the TM4C123 is configured to be compatible with LCD. Finally, in the NOKIA 5110 configuration part, the display is initialized for communication and to receive data to be displayed.

The Nokia 5110 uses SPI signals to receive commands, text, and images to be displayed. As to be noticed in Figure 4 on the SPI signals that there is only one data output signal. The LCD somehow, needs to distinguish whether the data being sent is data meant to be displayed, or if it is a command meant to control the screen. This is done not through the SPI module, but "manually" through a GPIO pin. The Nokia screen has a pin named Data/Command (DC). When the DC signal is low, the Nokia interprets the incoming SPI bits as a command. Similarly, if the DC signal is high, the SPI bits are interpreted as data to be displayed. The DC signal can be set high or low long before the last bit is sent.

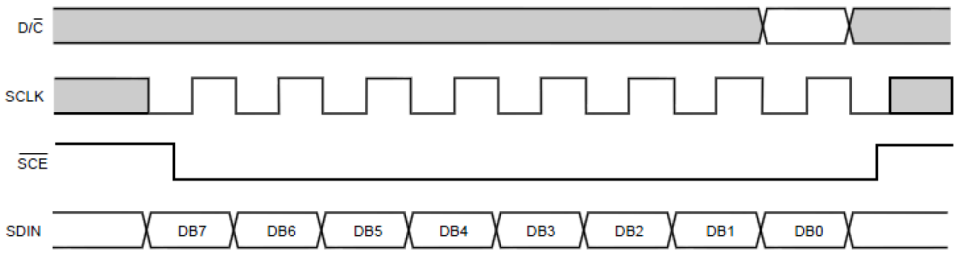


Figure 4: How the DC signal is timed with the SPI signals

For GPIO and SPI configuration parts, you may refer to page 965 of the TM4C123 Datasheet for details and 967 for register map.

GPIO Configuration:

1. Enable the clock for GPIOx **(RCGCGPIO)**
2. Wait until GPIOx is ready **(PRGPIO)**
3. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) pins as a digital pin **(DEN)**
4. Set directions for the pins **(DIR)**
5. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) pins for their alternate function **(AFSEL)**
6. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) port control pins to route the SSI interface to the pins **(PCTL)**.

SPI Configuration:

1. Enable the clock for SSIx **(RCGCSSI)**
2. Wait for the SSI peripheral to be ready **(PRSSI)**
3. Disable the SPI interface **(CR1)**
4. Set the clock rate of the SPI Clock **(CPSR, CR0)** accordingly. Please mind the maximum data rate that the LCD is capable of working with.
5. Set the data size to be 8-bits and Freescale mode **(CR0)**
6. Set the SPI mode **(CR0)** accordingly.
7. Re-enable the SPI interface **(CR1)**

For Nokia 5110 LCD configuration you may refer to Nokia5110Datasheet.pdf provided along with the project manual, where a detailed version of the instruction set and data transmission graphs are provided for clarification.

NOKIA 5110 Configuration:

NOTE: Please do not try to connect and turn on the backlight of the display early on in development. It requires 5V which is different than the Vcc of 3.3V. Instead, make sure you can correctly drive the display without the backlight. Even then, the backlight is still optional, so you may not connect it at all if the content of the display is visible under normal conditions.

1. To initialize the Nokia screen first toggle the Reset pin by holding it low for 100ms then setting it high.
2. Send the following commands to initialize the display
   - Set H=1 for Extended Command Mode, V=0 for Horizontal Addressing
   - Set V_{OP}. You may need to sweep values between 0x[B0-C0] for correct operation.
   - Set temperature control value. You may need to sweep values between 0x[04-07] for correct operation.
   - Set voltage bias value as 0x13.
   - Set H=0 for Basic Command Mode
   - Configure for Normal Display Mode
   - Set Cursor to detemine the start address
3. Send data to be displayed.

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

Figure 5: Instruction Format

It is possible to write data into the address of memory (DDRAM) of the Nokia LCD continuously and values of X-Address and Y-Address will be increased automatically. In this case, there are 2 methods to configure the operation format of address; firstly, Vertical Addressing Mode (V=1), 1 value of Y-Address will be increased every time; and secondly, Horizontal Addressing Mode (V=0), 1 value of X-Address will be increased every time. One may use either addressing as pleased; however it is to be noted that when the cursor is set to an arbitrary location, a very short delay, in the order of nsec is required for the cursor to settle.

5 Deliverables

You are supposed to attempt the project work individually.

- **Final Report:** A full description of your work, including photos of your setup and fully executable codes (printed). Please clearly indicate how you fulfill the requirements by satisfying the restrictions. Moreover, your codes should be well-commented. **Deadline: Jan 31, 2021**
- **Source Code:** A zipped Keil μvision project folder with fully executable codes is to be uploaded to ODTUClass. **Deadline: Jan 31, 2021**
- **Lab Demo:** Demonstration of the operation of the project. **On Feb 1, 2021**

References

[1] TI, "Tiva™ c series tm4c123g launchpad evaluation board user's guide." http://www.ti.com/lit/ug/spmu296/spmu296.pdf.

[2] TI, "Tiva™ tm4c123gh6pm microcontroller data sheet." http://www.ti.com/lit/ds/spms376e/spms376e.pdf.