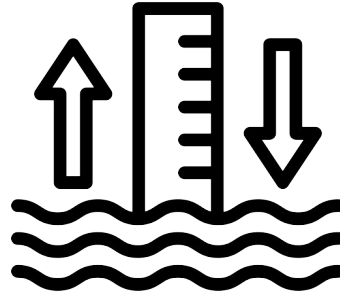




**ELECTRICAL AND ELECTRONICS  
ENGINEERING  
DEPARTMENT**

**Laboratory Project:**  
Water Level Controller



## Laboratory Project - Water Level Controller

### Objectives

In the EE447 laboratory work, you were expected to familiarize yourself with the operation of TM4C123G and its utility modules. 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 setup
- Introduction to 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 water level controller system based on an obtained water level signal. You are going to detect water level using a water level sensor, and try to keep the water level in a range using two water pumps. To show the current configuration and measurements, you will use a Nokia 5110 LCD, which you will drive using SPI module. You will also use GPIO for the on-board RGB LED and pushbuttons.

The system has three major functions:

## 1.1 Water Level Sensing

The system continuously senses the water level using a sensor. It stores samples in an array of 256 elements. When the array is filled, your system calculates the average water level and updates the output elements' states.

## 1.2 Controlling Water Level

The system tries to keep the water in a predefined range using two pumps: filling-pump and discharge-pump. While filling-pump fills the container with water taken from another container, discharge-pump is used to drain water in the container.

## 1.3 User Interface

The system has three output elements. Firstly, on the LCD screen, detected water level and the water level range are displayed. Secondly, on-board LEDs are turned on or off according to the detected water level. If the water is below the range, red LED must be on and the others must be off. If the water is in the range, green LED is on. And blue LED must be on, when the water level exceeds the range.

## 2 Requirements and Restrictions

The overall functionalities of water level controller system are described in Section 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 water level sensor, Nokia 5110 LCD screen, two water pumps and on-board LEDs as the user interface. You may also use a potentiometer or a keypad for threshold setting as a bonus. Following are the functional requirements:

1. The system should have one constant (see item 8) range (i.e. low and high range limits).
2. If the water is below the range, red LED must be on and the others must be off.
3. If the detected water level is in the range, green LED should be on.
  - **(BONUS)** When a green LED is on, its brightness can change proportional to the current water level. That is the LED should lights up less in low water and more in high water levels.
4. If the detected water level exceeds the range, blue LED should be on.
5. The user should be able to see the configured range on the screen.
6. The user should be able to see the current water level on the screen.
7. The user must set the range (low and high limits) using a potentiometer. For this, you need to read the potentiometer using ADC channel.
8. **(BONUS)**
  - Optionally, you may set the range by entering decimal digits using the 4x4 keypad. In this case, the user should press a button on the keypad to enter threshold setting mode. A second press to that button will set the threshold and exit the threshold setting mode.
  - You may display detected water level as a horizontal dot plot graph (time vs water level) on the LCD screen.

## 2.2 Restrictions

### 2.2.1 Hardware Restrictions

The uses the following components:

1. Water Level Sensor
2. NOKIA 5110 LCD Screen
3. 1 Potentiometer (optional)
4. 1 4x4 Keypad (optional)
5. RGB LED Placed on the TM4C123G Board
6. 2 Water Pumps

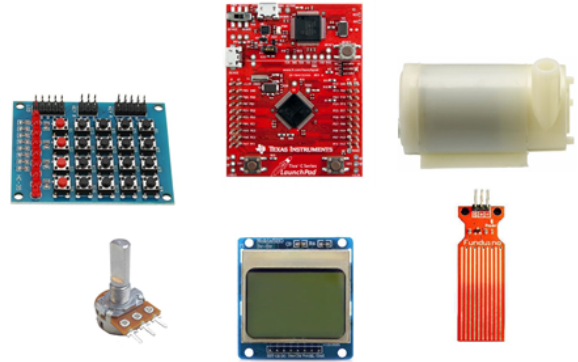


Figure 1: Components

### 2.2.2 Implementation Restrictions

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

1. Water level sensor must be read using the ADC module.
2. 256 samples should be collected to calculate the detected water level.
3. The water pumps must be driven using a transistor.

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.

### 2.2.3 Programming Language Restrictions

- In order to program Nokia 5110 LCD screen, only ARM assembly language must be used. That is, calling any other C function/subroutine etc. is strictly forbidden.
- C programming language will be used for the remaining sections.

### 3 Tips

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 [1]) to be configured. You can find the interrupt number of an interrupt source from the table in page 104 of [1].

Under the given restrictions, you will read the ADC using SysTick interrupt handler and drive the water pumps 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. Wait for ADC module to store 256 readings in the array
2. Calculate the average
3. Turn on/off the corresponding LEDs and drive the corresponding water pump
4. If more than a second has elapsed after the last LCD update, show the water level value on LCD
5. Loop back to 1.

**Note:** For detailed information about I2C module in TM4C123 see page 997 of [1].

**Note:** In order to test the system, the water will be added to or drained from the container.

**Note:** Time domain specifications of a system - rise time, peak time, settling time etc. - are not going to be considered while evaluating the project.

## 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.

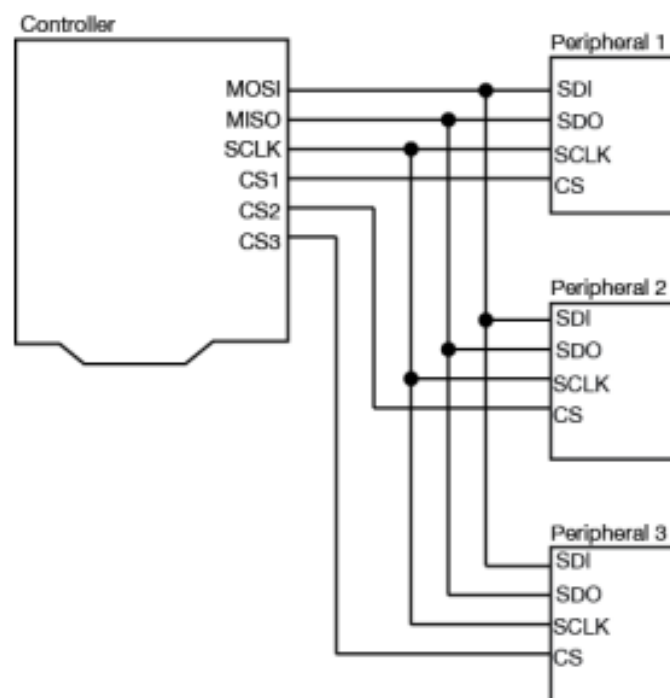


Figure 2: Synchronous Serial Interface in the TM4C

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

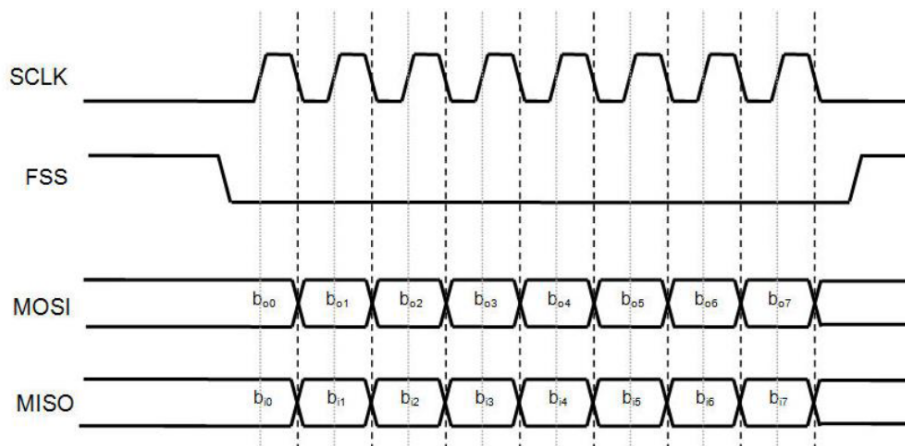


Figure 3: How SPI signals change as data is sent

## 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.

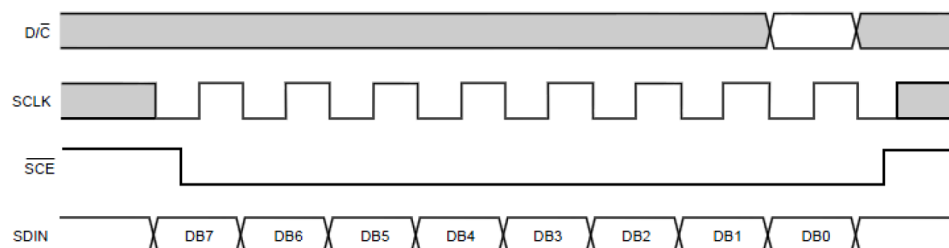


Figure 4: How the DC signal is timed with the SPI signals

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 determine the start address
3. Send data to be displayed.



INSTRUCTION	D/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 <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	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 <sub>2</sub>	Y <sub>1</sub>	Y <sub>0</sub>	sets Y-address of RAM; 0 ≤ Y ≤ 5
Set X address of RAM	0	1	X <sub>6</sub>	X <sub>5</sub>	X <sub>4</sub>	X <sub>3</sub>	X <sub>2</sub>	X <sub>1</sub>	X <sub>0</sub>	sets X-address part of RAM; 0 ≤ X ≤ 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 <sub>1</sub>	TC <sub>0</sub>	set Temperature Coefficient (TC <sub>x</sub> )
Reserved	0	0	0	0	0	1	X	X	X	do not use
Bias system	0	0	0	0	1	0	BS <sub>2</sub>	BS <sub>1</sub>	BS <sub>0</sub>	set Bias System (BS <sub>x</sub> )
Reserved	0	0	1	X	X	X	X	X	X	do not use
Set V <sub>OP</sub>	0	1	V <sub>OP6</sub>	V <sub>OP5</sub>	V <sub>OP4</sub>	V <sub>OP3</sub>	V <sub>OP2</sub>	V <sub>OP1</sub>	V <sub>OP0</sub>	write V <sub>OP</sub> 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 in group of two. Groups will be determined by course instructors and teaching assistants.

- **Preliminary Report:** A summary of the framework (including flowcharts and pseudo codes), at most 3 pages. **Deadline: 25.12.2022**
- **Final Report:** A full description of your work, including photos of your setup and relevant portions of code you may need to explain certain key points. Full source codes are not required in your report. Please clearly indicate how you fulfill the requirements by satisfying the restrictions. Moreover, your codes should be well-commented. **Deadline: 08.01.2023**
- **Source Code:** A zipped Keil  $\mu$ vision project folder with fully executable codes is to be uploaded to ODTUClass. **Deadline: 08.01.2023**
- **Lab Demo:** Demonstration of the operation of the project. **Date: TBA**

## References

- [1] TI, “Tiva™ tm4c123gh6pm microcontroller data sheet.” <http://www.ti.com/lit/ds/spms376e/spms376e.pdf>.