Deniz Karakay – 2443307
Hüseyin Totan – 2446979 25.12.2022

EE447 – Project Preliminary Report for Water Level Controller

Introduction

We aim to construct water level controller system using water level sensor, water pumps, Nokia 5110 LCD, potentiometer, on-board RGB LED and external push-buttons. We sense the water level using the sensor and control pumps accordingly. We will use on-board RGB LED to indicate the status of the system and LCD to display the limits and status of the sensor. Potentiometer and push-buttons will be used to modify the upper and lower limits.

Water Level Sensor

This water level sensor has 5 sense and 5 power traces to determine the level of the water. Normally these traces are not connected, but in the water, they are bridged. The sense and power traces create a variable resistor just like a potentiometer whose resistance changes depending on the water exposure level. The resistance of the sensor changes inversely proportional to the level of water exposure. In other words, if we put the sensor in a place where the water level is high, the resistance decrease. The sensor produces an output voltage that is proportional to the resistance. We can measure this voltage to determine the water level. You can examine the pseudo code to enable pins and retrieve data.

```
void init_sensor_pins(){
Enable Port E
Enable ADC0

Use GPIO PE3
Enable AFSEL to use as ADC
Set direction and disable digital enable
Enable AMSEL

Disable Sequencer 3
Clear 12-15 pins of EMUX
Sample Sequence Control 3 Set IEN0 & END0

Select 125 ksps
Enable ATD

}

volatile float get_sensor_data()
introduce volatile float variable
check RIS flag for Sequencer 3
        if they are set
                Read data using SSFIFO3
        else
                Do nothing
Return data
```

LCD Control

The following pseudo codes illustrates our solution approach towards the display problem. In the application an equivalent assembly code will be written to provide such interface to the main program. The LCD part of the code will receive only upper limit, lower limit and current level parameters from the main program.

```
void AFSEL(portA,pints2,3,5, output, afsel=2);  //these 2 functions are available to us from
void GPIO_INT(portA,pin4, output);              //previous assignments
```

```
void send_command(char command);
void LCD_int(){
	GPIO_INT(portA,pin4, output); //set portA pin4 for D/C select of lcd device
	enable clock of SSI0;
	short_delay;
	disable SSI0;
	AFSEL(portA,pints2,3,5, output, afsel=2);	//set necessary SSI0 pins;
	set size and data type of SSI0;
	set clock divisor of SSI0;
	enable SSI0;
	short delay;
	send_command(0x21);  //enable advanced command set
	send_command(0x14);  //set Vop to 3.3V
	send_command(0x07);  //set temp coefficient
	send_command(0x13);  //set bias
	send_command(0x20);  //disable advanced command set
	send_command(0x0C);  //set normal display mode

}
void send_command(char command){
	PortA_address[(0x10)<<2]=0;   //set portA pin4 to 0
	SSI0[SSIDR]=command;			//send command to SSI0 fifo
	while((SSI0[SSISR]&1)==0);    //wait fifo to empty
}
void send_data(char data){
	PortA_address[(0x10)<<2]=1;   //set portA pin4 to 1
	SSI0[SSIDR]=command;			//send command to SSI0 fifo
	while((SSI0[SSISR]&1)==0);    //wait fifo to empty
}
void LCD_WRITE(char* buffer){
	 send_command(0x80); //set x to 0
	 send_command(0x40); //set y to 0
	for(int y=0;y<6;y++){
		for(int x=0;x<84;x++){
			send_data(buffer[84*y+x]);
		}
	}
}


const static int Cy,Uy,Ly,off;	//will be determined experimentally

void LCD_DISP(int lower_l, int upper_l, int current_l){
	 LCD_WRITE(static_background);
	send_command(0x80&off);			//set x
	 send_command(0x40&Cy);			//set y
	for(i=0;i<current_l;i++) send_data(0x1F);
	send_command(0x80&off);			//set x
	 send_command(0x40&Uy);			//set y
	for(i=0;i<upper_l;i++) send_data(0x1F);
	send_command(0x80&off);			//set x
	 send_command(0x40&Ly);			//set y
	for(i=0;i<upper_l;i++) send_data(0x1F);
}
```

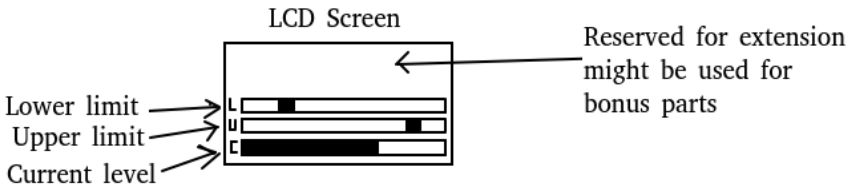

Figure 1: LCD Screen Display template overview

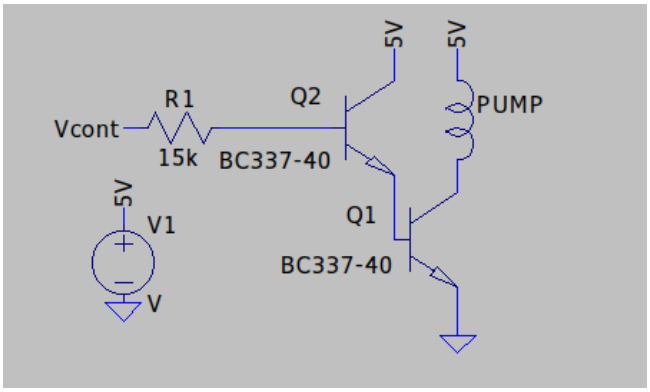

Figure 2: Pump switching circuit

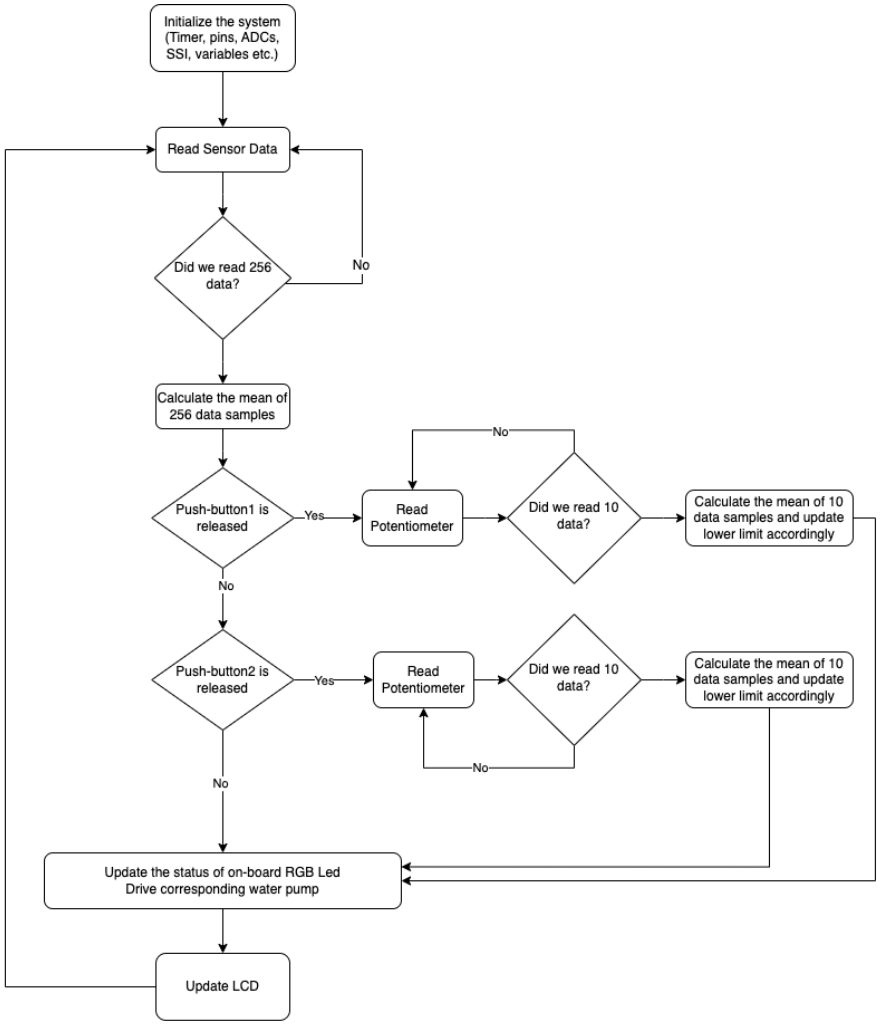

Figure 3: The System Flowchart