Lab 3 – Prototype Phase 1

ECE 298 – W2021

Lab Section: | Group: 34

Done by Shunethra Senthilkumar, Rajan Sood

Part 1 – Pin Mapping

Table 1: MCU pin use

MCU Pin	Pin Mode	Functional Description
PA1	GPIO_EXTI1	Handles the interrupts for the position limit sensor.
PA6	TM3_CH1	Send PWM trigger for the ultrasonic sensor module.
PA8	GPIO_Output	Send Red LED signal to indicate door motor is in motion.
PA9	GPIO_Output	Send Yellow LED signal to turn on to indicate collision state.
PA10	GPIO_EXT10	Handles the interrupts for the collision sensor.
PA11	GPIO_Output	Send Green LED2 signal to turn on if there is an object detected within P1 distance.
PA12	GPIO_Output	Send Green LED1 signal to turn on if there is an object detected within P2 distance.
PB0	GPIO_Input	Receives signal to (open the door) or (increase a parameter) according to the press and release of pushbutton.
PB1	GPIO_Input	Receives signal to (close the door) or (decrease a parameter) according to the press and release of pushbutton accordingly.
PB4	GPIO_Input	Receives signal indicating selection of Setup mode operation.
PB5	GPIO_Input	Receives signal indicating selection of Run mode operation.
PB6	TIM4_CH1	Generates PWM for one of the motor inputs to ramp up and down the motor.
PB7	GPIO_Input	Receives signal when parameter P2 is selected in setup mode.
PB8	TIM4_CH3	Generates PWM signal for the other motor input to ramp up and down motor.
PB10	GPIO_Input	Receives signal when parameter P5 is selected in setup mode.
PB12	GPIO_Input	Receives signal when parameter P3 is selected in setup mode.
PB14	GPIO_Input	Receives signal indicating selection of Locked mode operation.
PC3	GPIO_Input	Receives signal when IDX output of the motor goes high, helps to determine number of revolutions of motor.
PC6	USART6_TX	Handles communication services to the LCD module.
PC7	USART_RX	Handles data transmission services to the Virtual terminal.
PC8	GPIO_EXT18	Handles interrupt from ultrasonic sensor for outside distance (P1)
PC9	GPIO_EXTI9	Handles the interrupt from ultrasonic sensor for inside distance(P2)
PC10	GPIO_Input	Receives signal when parameter P1 is selected in setup mode.
PC11	GPIO_Input	Receives signal when parameter P4 is selected in setup mode.

Part 2 – MCU Resources

Table 2: MCU resource use

MCU Resource	Functional Description
USART6	Communicates with the LCD screen module
TIM4	Generate PWM signal for DC motor driver FETs
TIM2	Used for measuring the ECHO PW.
TIM3	Generate PWM signal for the ultrasonic sensor.
GPIO	Used for various inputs and outputs for LEDS, switches, etc.
NVIC	Used for the interrupts from ultrasonic sensor, collision switches, position limit switches.

Part 3 – Schematics and Test Cases

User inputs:

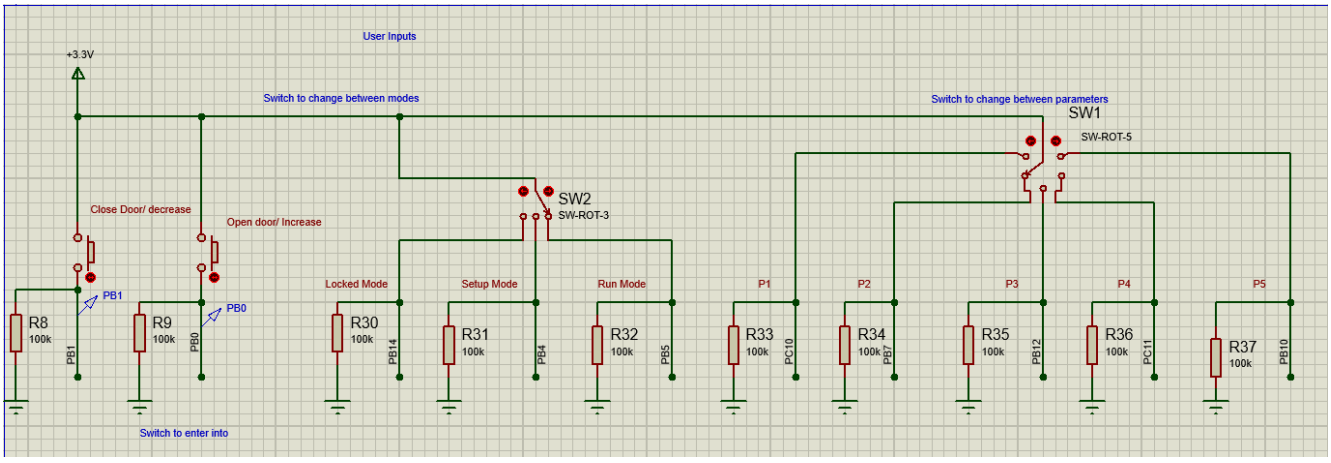
We use a combination of two pushbuttons, SW2 (3 point switch) and SW1 (5 point switch) for the user to interact with the system. We used pull down resisters across each to get rid of any floating voltages and prevent sending "accident" signals to the MCU.

The user can switch between different modes using the SW2.

When the user is in the setup mode they can use SW1 to select the desired parameters (P1, P2, P3, P4, P5).

- When in Locked mode, the user can click the PB1/PB2 to either open/close the door.
- When in the setup mode and the desired parameter is chosen, we can press PB1 or PB2 to either increase/decrease the parameter value.
- For P4 and P5 we chose the ramp up/ramp down constants as: slow (P4/P5 = 0.5) , normal(P4/P5 = 1), and fast (P4/P5 = 2). By default the system starts in normal mode.
- When in Run mode the user can press the pushbutton at the ultrasonic sensor to "trigger" (a PWM is being generated by the MCU at PA6 and closing the switch would simply pass the PWM to the ultrasonic sensor) the ultrasonic sensor.

Schematic:

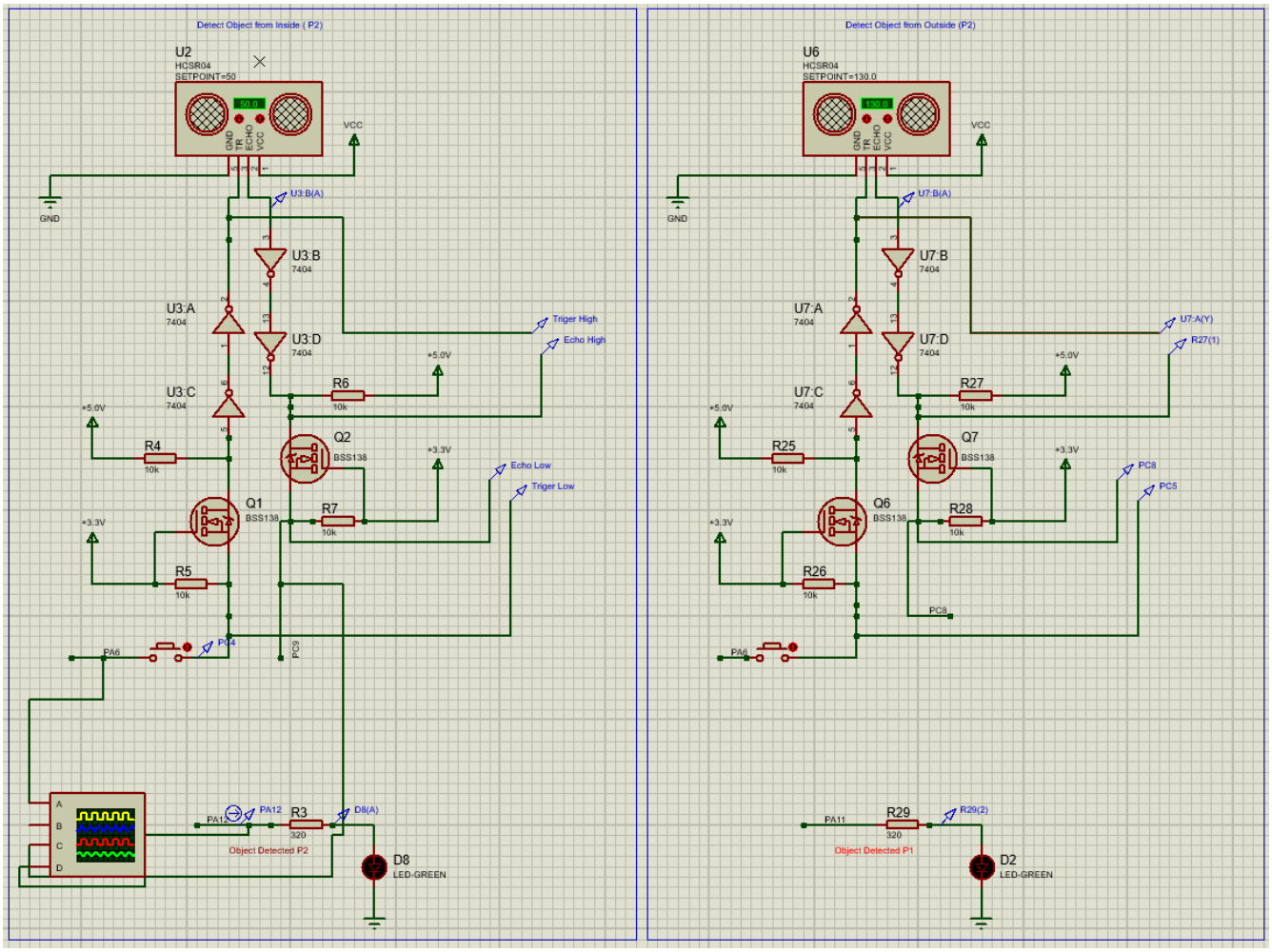


ULTRASONIC SENSOR

Ultrasonic Sensor Schematic:

We are using the same ultrasonic sensor schematic from previous Lab, however we use a PWM signal generated from the MCU at PA6 to trigger the ultrasonic sensor. We can press the pushbutton (located just after PA6 pin) to pass the PWM to the ultrasonic sensor to simulate an object has been detected. We use the Green LED D8, D2 to indicate we have detected an object at P2 /P1 respectively.

- We turn off the respective LED after we have fully opened and closed the door.
- While the door is closing and we press the pushbutton to simulate an object has been detected at P1/P2 we will stop closing the door and move to opening the door (once the door is fully opened we will begin closing the door after the user set time “P3”).



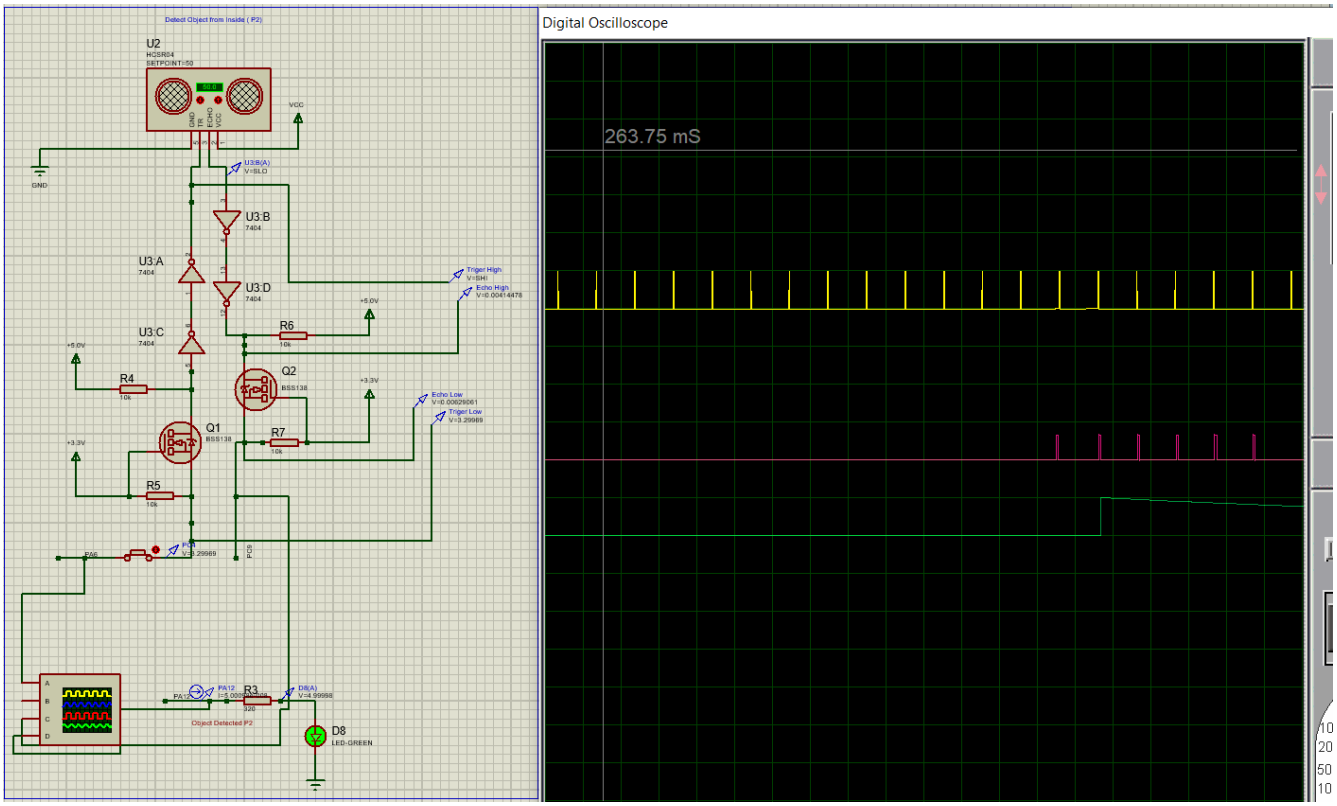
Ultrasonic Simulation:

We have PA6 on channel A, the echo output at CH C and LED on CH D.

We can observe the PWM being generated by the MCU and when we press the pushbutton, we receive an output at the echo which triggers an interrupt to the MCU via PC9/PC8. In our interrupt handling function whenever we observe an interrupt at PC9/PC8 we start a timer2 and we stop the timer when the PC9/PC8 == PIN_STATE_RESET hence we basically have the ECHO PW time which we can use to calculate the distance using the formula (ECHO PW time/ 58 = distance cm).

We can then check if the distance measured lies within the distance set by user using the parameters P1/P2.

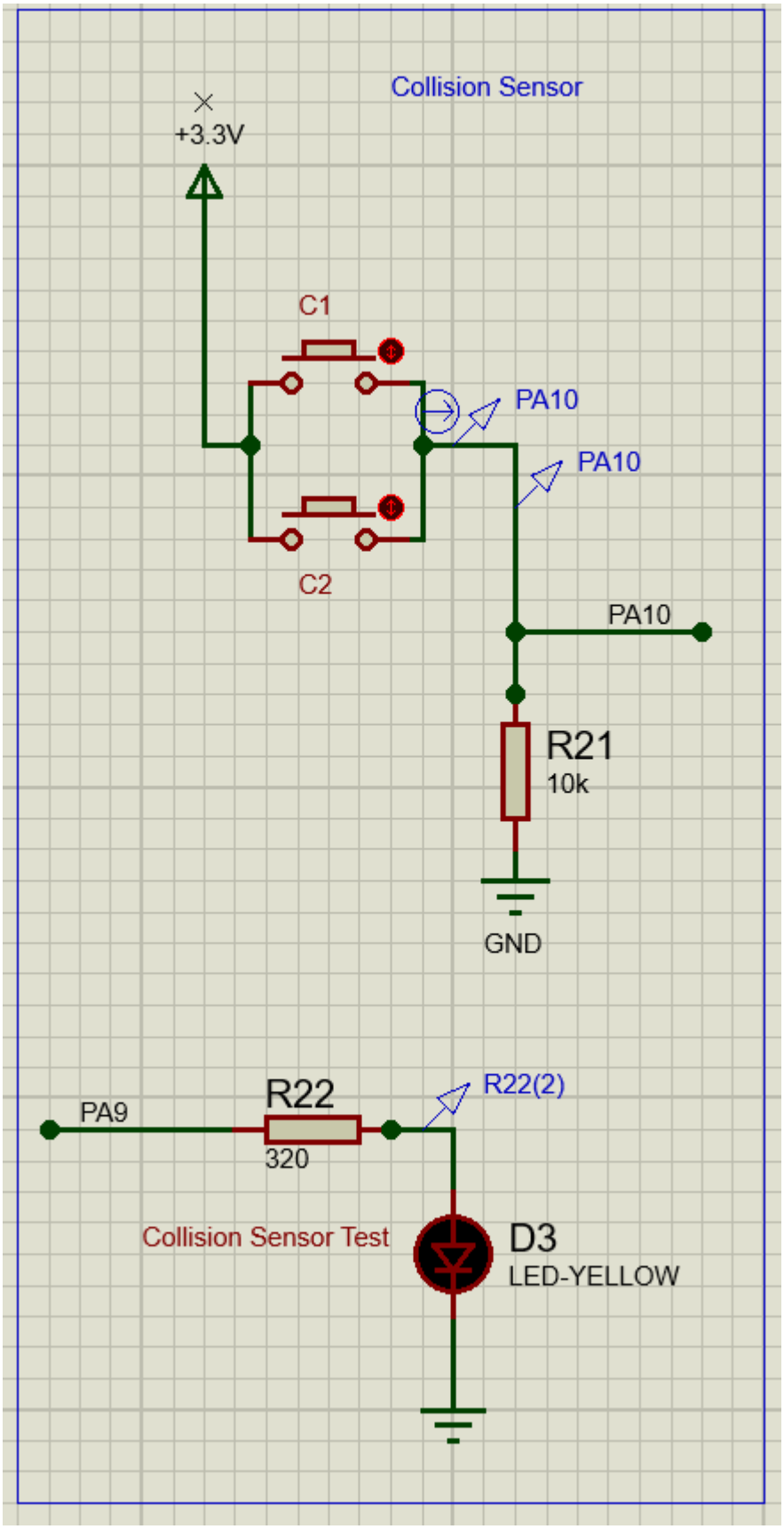
If the measured distance lies within the set distance we can SET the respective LED and set the Flag that we have detected an Object and we should start opening the door (we call the openDoor() function).



COLLISSION DETECTION

Collision occurs when MCU pin PA10 receives a signal that collision has occurred. The 0->1 transition on PA10 causes an interrupt, thereby signalling that a collision has occurred to the MCU. When the collision is detected through PA10, the pin PA9 is toggled to implement a flashing Yellow LED.

Schematic:


Collision Sensor
+3.3V
C1
PA10
PA10
C2
PA10
R21
10k
GND
PA9
R22
320
R22(2)
Collision Sensor Test
D3
LED-YELLOW

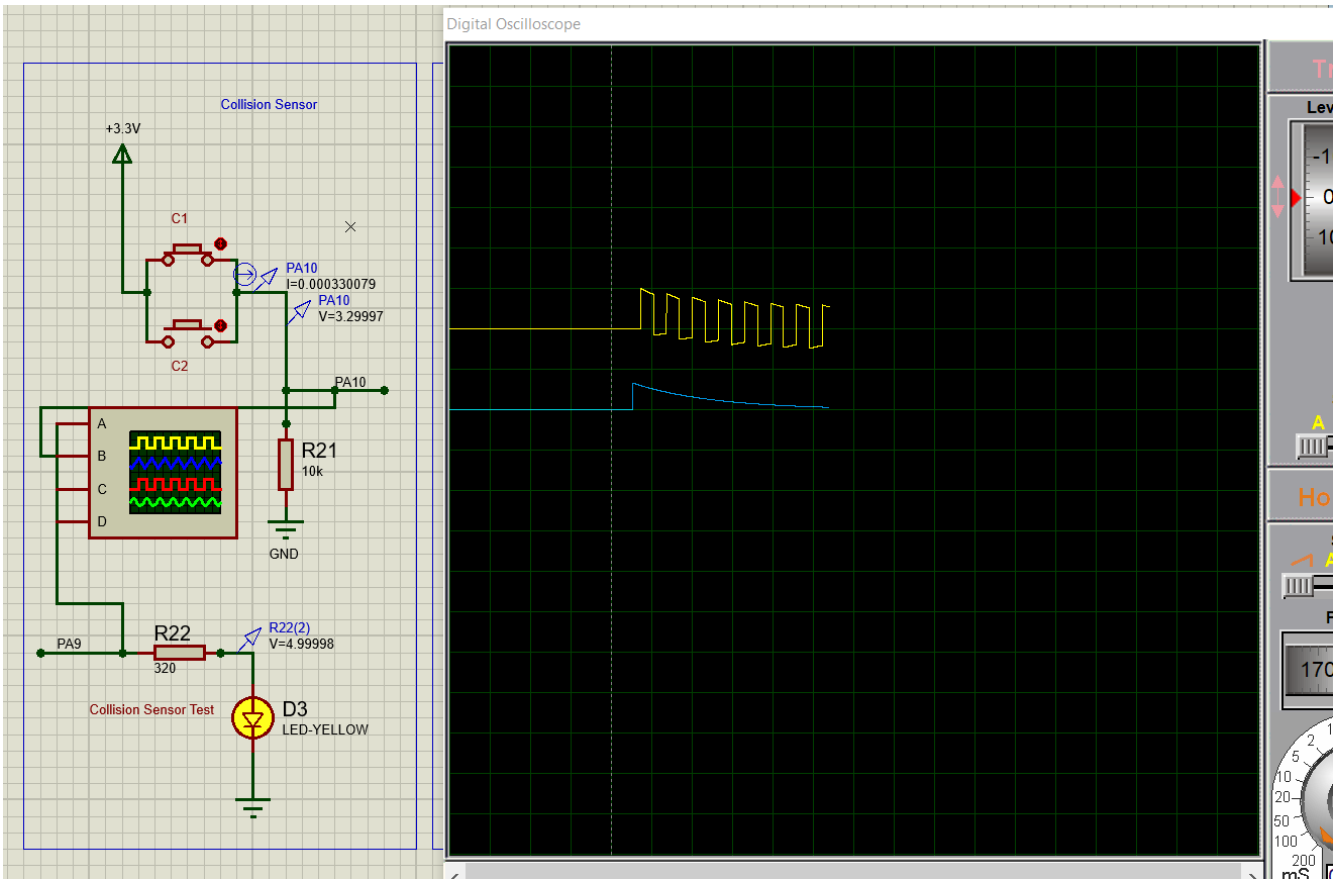
Simulation:

C1 &C2 Not Pressed (no collision):

- Initially push button C1 and C2 are not pressed (meaning that no collision has occurred). As there is no collision, PA10 receives zero voltage and there is no interrupt caused. We can observe this zero voltage as shown by Channel B of the oscilloscope.
- As there is no collision detected, pin PA9 is also not set by the MCU. This is seen as zero volts in Channel A of the oscilloscope.

C2 Pressed (Collision detected):

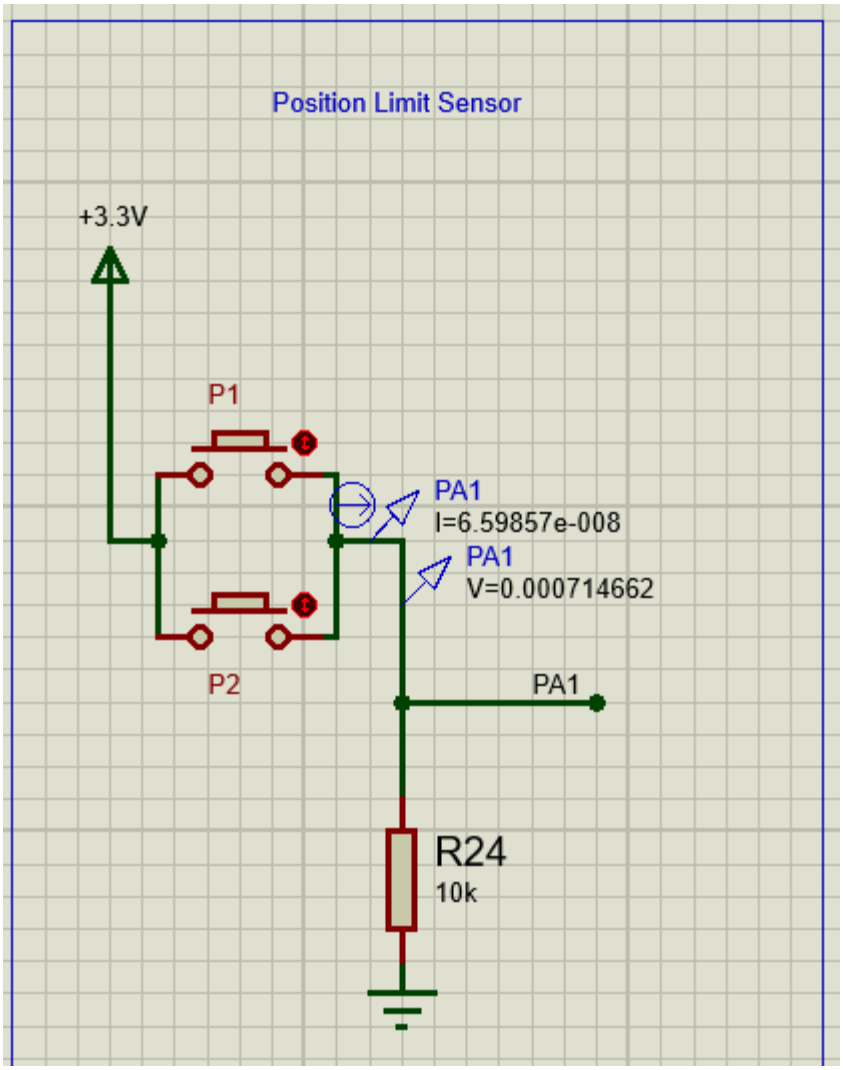
- When any one of the collision buttons are pushed, a collision occurs in our system. Here we simulate the condition where C1 is pressed. when the push button C1 is pressed (meaning that collision has occurred on one side), a hardware interrupt occurs making PA10 go high. This is indicated in Channel B (blue colour) of the oscilloscope.
- As interrupt is detected, we can see that pin PA9 toggles to make the Yellow LED go on and off leading to a flashing LED pattern. This is shown in Channel A of the oscilloscope - we see continuous pulses only when C1 is pressed.



DOOR POSITION LIMIT SWITCHES

Door Position limit switches stop the door motion when the door reaches the minimum and maximum travel limits. When one of the limit switches in pressed, an interrupt is generated through pin PA1. The status of this interrupt signal will be used to stop the motor power accordingly.

Schematic:



Simulation:

P1 & P2 Not Pressed:

This shows the case when the door has not reached its minimum or maximum limits. Hence, the DC Motor operates with the set revolutions and speed in the properties. Pin PA1 outputs zero in this case as there is no occurrence of interrupt here.

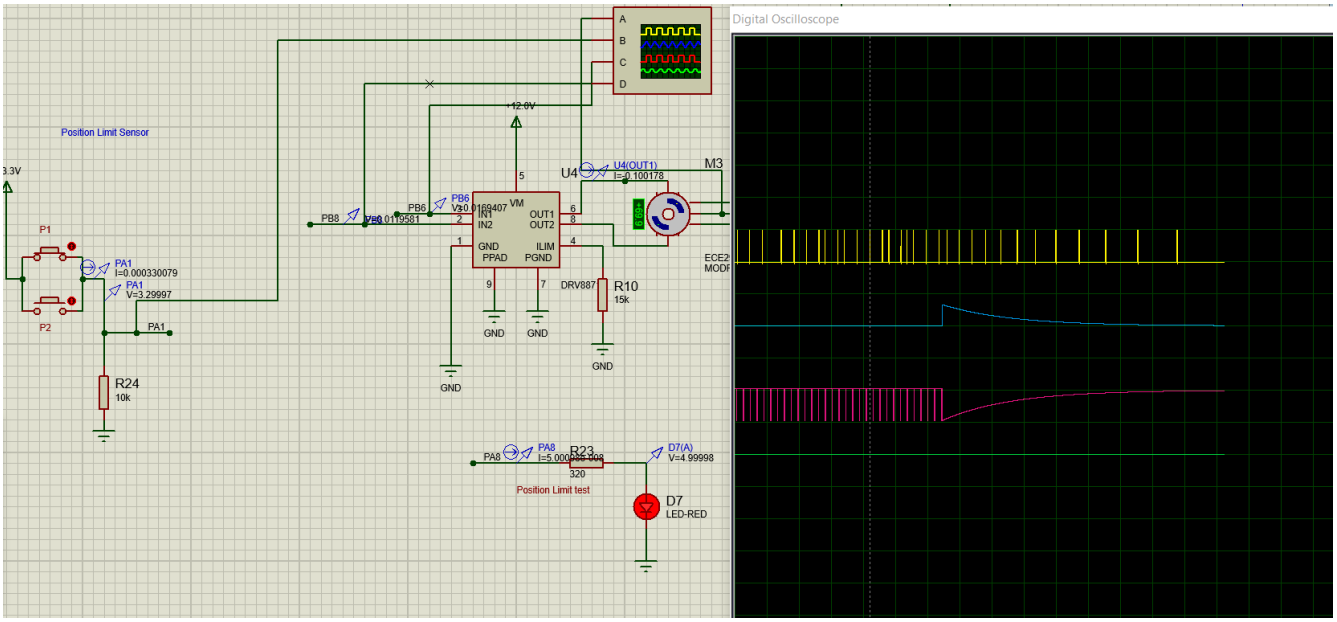
P1 Pressed:

This shows the case when the door reaches its set limit points. We simulate this condition by pressing pushbutton P1 to indicate that maximum/minimum allowable limit has reached.

Motor IDX output (Channel A): When P1 is pressed, the door has reached its limit. The motor power is disabled here. This is indicated by Channel A in the oscilloscope – the IDX outputs occur after a longer duration compared to the initial stage. Hence, we can verify that the motor power is being disabled leading to the stopping of the motor.

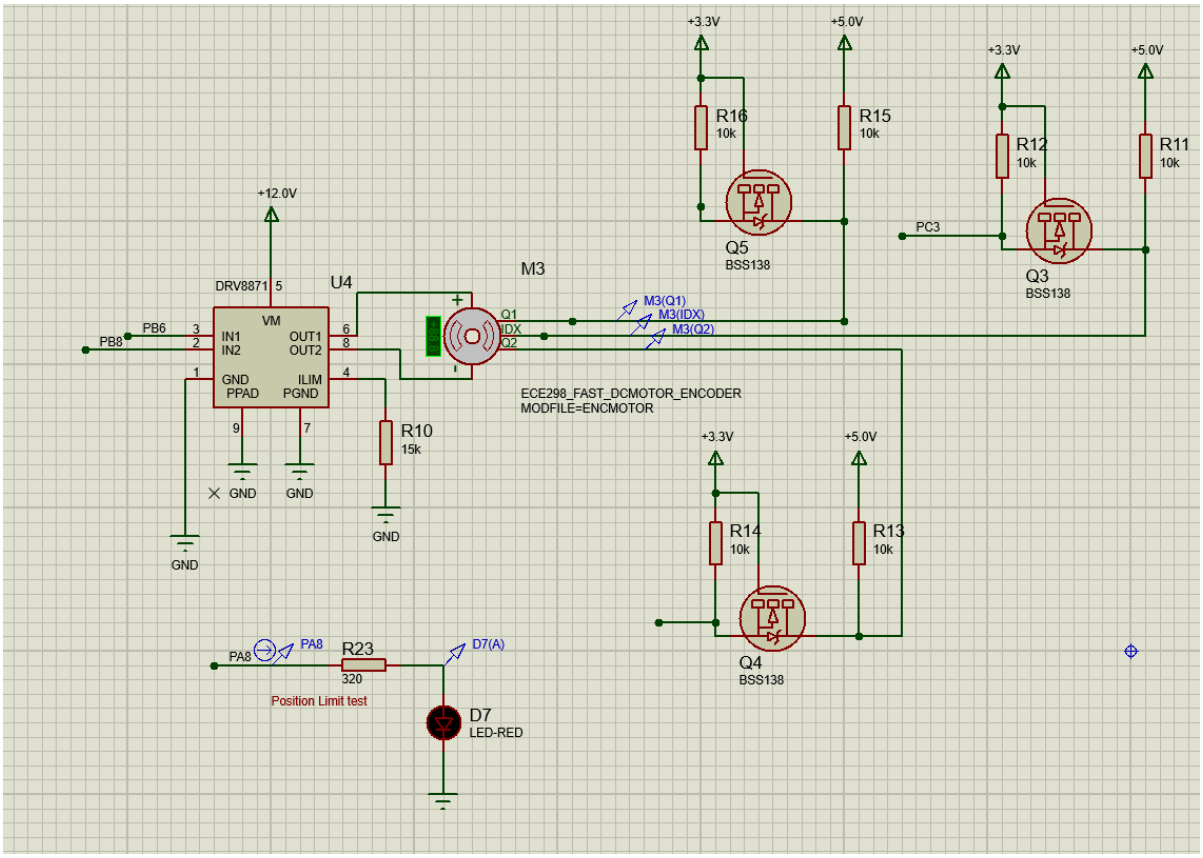
Interrupt pin PA1 (Channel B): When P1 is pressed, a hardware interrupt is received by PA1 MCU pin. The PA1 pins goes high as soon as P1 is pressed as seen in channel B of the oscilloscope.

Motor Inputs (Channel C & D): When P1 is pressed, we can see how there is a change in the PWM input signals to the motor. The PWM signal just goes to 1 without any cycles like before.



DOOR MOTOR

Motor Schematic:

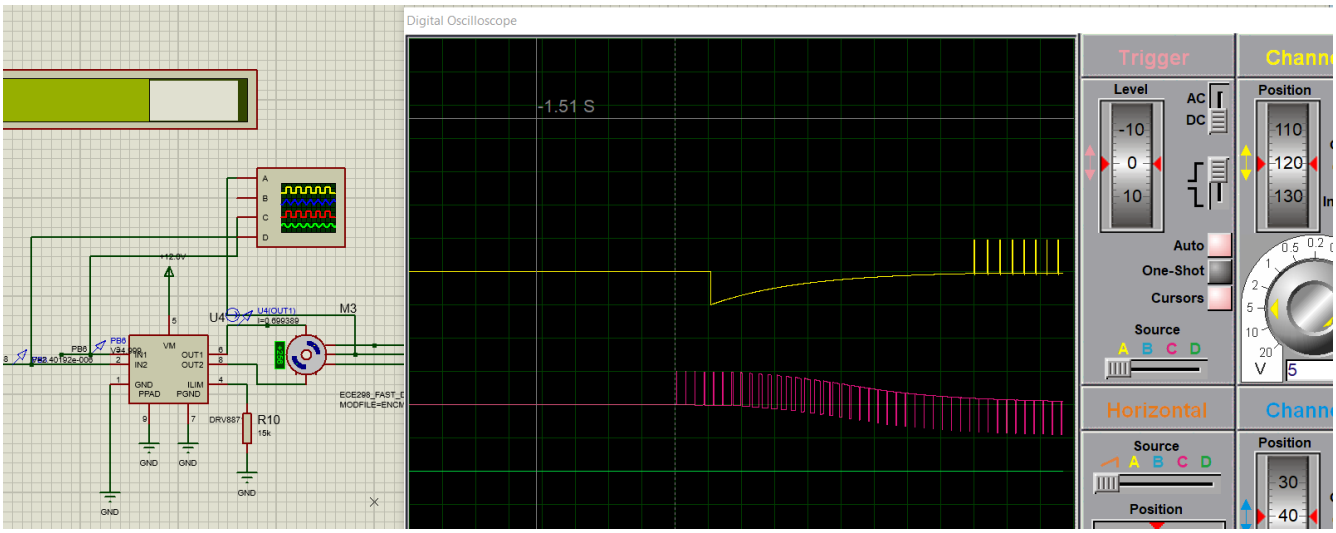


Motor ramping up Simulation:

Whenever we call the openDoor function in the code, we start opening the door by ramping up the motor to a steady value of ~390 rpm.

We are generating PWM on PB6 and PB8 via the MCU, while opening the door we first start generating the PWM on PB6, we control the ramping up changing the PW of the PWM and effectively changing its duty cycle as such:

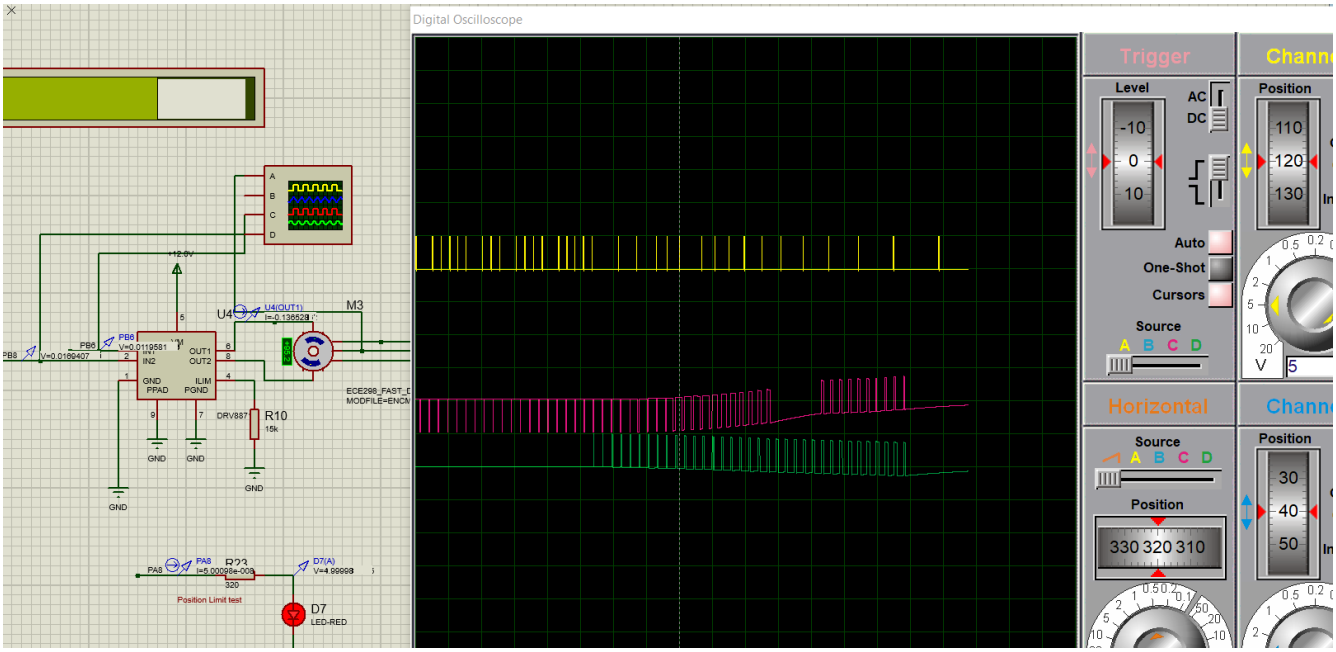
- We set our counter period to be 1000.
- We start PWM with a PW of 50 and then increase the PW of the PWM in while with increment of (PW_OLD + (50*P4)) (where P4 = 0.5 or 1 or 2 depending on what the user selected in setup mode) every ~80ms to a max of 980.
- By setting the max to be 980 which is essentially 98% of max RPM. Hence we will achieve the steady state value of 390 rpm.
- Ramping up in such a fashion helps us to avoid to high inrush current and stay below 1.2A



Motor ramping down Simulation:

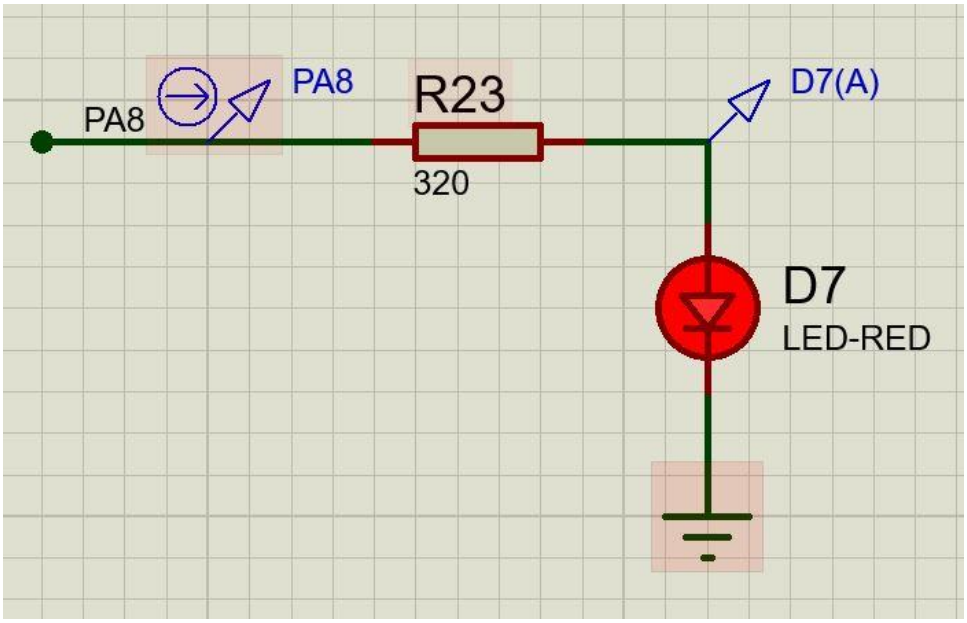
As we are keeping count of the IDX “highs” we know the speed and position of door during the opening/closing process. Hence when are roughly 1ft away from closing we would have completed roughly 1050 revolutions have 150 more revolutions to complete.

- in the while loop we check if we have completed 1050 revolutions and if we have start generating the PWM on PB8 (as seen on the scope).
- PWM on PB8 is started with an initial PW of 50 and we increment its value by (PW_OLD + (50*P5)) (where P5 = 0.5 or 1 or 2 as selected by the user) every 2 revolutions.
- While we are increasing the PW at PB8 we are simultaneously decreasing the PW at PB6 in similar fashion.
- This method of controlling the PWM’s helps us achieve our ramp down for the motor and avoid smashing the door into the door frame and position switches.
- We have also added a fail safe in case there was an error in counting the IDX highs or we are approaching the end much faster than we should. If we are “50” revolutions away we stop supplying the PWM’s to the motor controller to ensure we are not slamming in and keep rotating the motor.



Door Motion Indication Simulation:

Whenever the door motor is in motion, we send a signal to toggle the PA8 pin to have red LED flashing. The LED doesn't flash when the door motion is stopped or disabled.



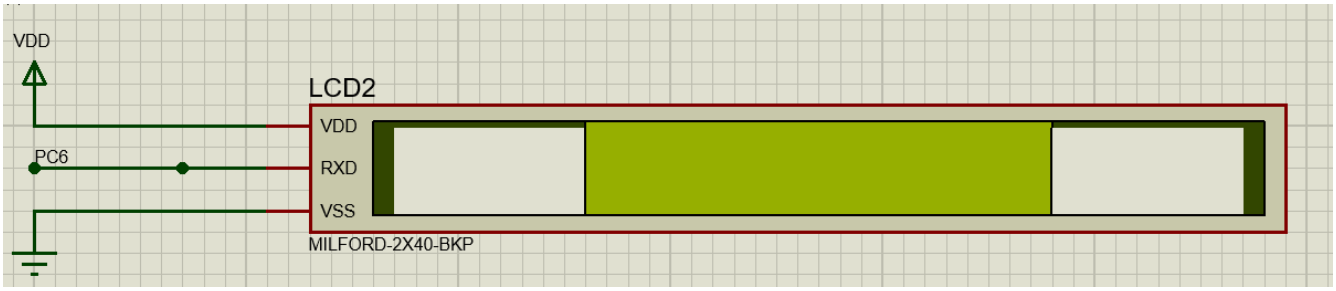
LCD:

schematic:

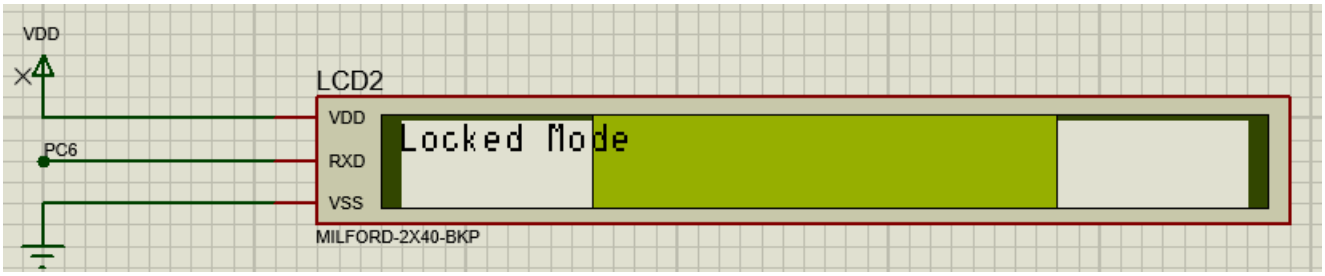
weird backlighting issue with the proteus component

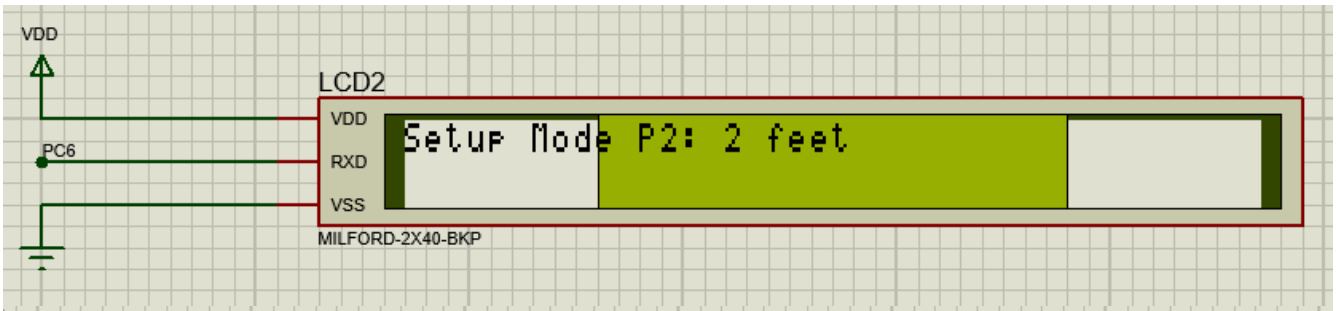
We used the MILFORD-2X40-BKP LCD for our system.

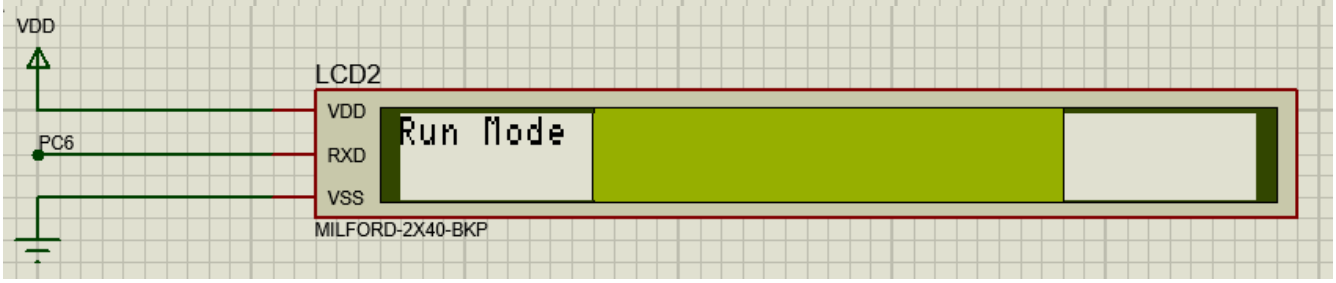
We Implemented the LCD using UART



Simulation:







Comprehensive System Testing:

Locked mode:

We tested the locked mode of the system by running the simulation and then using SW1 we choose “Locked Mode” (PB14). We confirm that the LCD is working as it its displaying “Locked Mode” (as seen in lcd simulation) we also make sure other switches and triggers (ultrasonic sensor) do not affect the system.

We the press PB1 (in the user input) to start opening the door, we observe that the motor starts ramping (as seen in motor ramp up simulation) up as we expected and the corresponding red LED starts flashing to indicate door movement.

- We also checked the collision state (as seen in collision simulation) by pressing the collision switch and we notice that the motor rapidly stops moving and the yellow LED starts flashing while we are holding the switch ON. When we let go of the switch indicating that the collision is resolved, we notice that the motor starts spinning again but we are limited to 200 rpm.

- We also observe that once we are ~1ft away from the door the motor starts to ramp down (as seen in motor ramp down simulation) to much a slower rpm. When we press the limit switch to indicate we have reached the end we observe that the signal to the motor is stopped and the motor is "disabled" (as seen in limit switch simulation).

*we check the door closing function in locked mode similarly.

Setup Mode

We then enter setup mode using SW1 and select "Setup Mode" (PB4). We check the LCD is working (as seen in lcd simulation) and we are in setup. We then testing changing different parameters using SW2 and selecting different parameters and try incrementing/decrementing their values.

- We observe that we are able to navigate through the different parameters and change their respective values using (PB1 and PB2)
- We also tried trigging the ultrasonic sensor, collision, limit switches in this mode to ensure there is no power to the motor and the system behaves as we expect it to.

Run Mode

We then enter the run mode using SW1 and select "Run Mode" (PB5). We check the LCD is working (as seen in the lcd simulation) we then simulate an object at the "inside" ultrasonic sensor using the PWM and pushbutton.

- We observe that there is an ouput at the echo pin and the interrupt is triggered. As the distance set on the sensor was within the P2 distance, this was a "valid" object and the motor starts moving to open the door. We observed the ramp up, collision, ramp down, limit switchs as we did in locked mode and ensured they were behaving as expected.
- After the door had fully opened and delay time (P3 set by the user) had elapsed, the motor starts to rotate in the -ve direction to close the door. We observe the same ramp up, ramp down characteristics.
- We also simulated an object approaching the door while its closing and we observe that the door stops closing and starts opening.
- We observed that all the systems behaved as we expected.

Possible source of error

- We noticed that there occasional random and irregular gaps in the IDX spikes,we tried with/without different codes and also direct signal from a generator and the issue was still present, which leads us to believe the error could be coming from the proteous simulation. This error can cause an incorrect "count" of spikes.