EE 128
Lab 5:Timer and PWM Operations

Section 021

TA: Mohsen Karimi

Tanish Arora
Leya Zeng

1. Abstract

This Lab's main objectives are to getting familiar with :

- Timer operations and pulse-width modulation techniques on both boards.
- To improve skills in designing and debugging microcontroller-based systems.

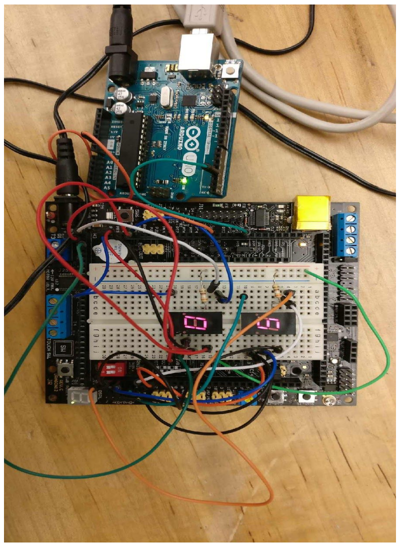
2. Procedure

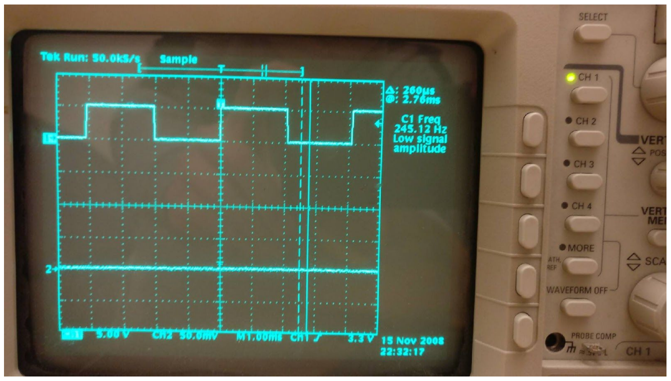
1) Get hands on training with the Arduino's PWM programming environment, Go to Files -> Examples -> Analog menu of the arduino software and find a "Fading" example which will actually demonstrate the use of analog output (PWM) to fade an LED which is basically turning LED on and off.
2) Run the code on lecture note to measure the duty cycle of a PWM signal coming into an input capture pin of the 9S12 microcontroller. Modify if necessary such that it measures the rise and fall edges and outputs it out to the 7 segment display.
3) Modify the Arduino source code such that the frequency of the PWM is 245Hz instead of 490 Hz. Verify that the 9S12 program still reads the correct duty cycle.
4) Make sure that the hardware is connected properly i..e, that the LED 7 segment display is connected according to the code to produce different outputs. And the same things goes for the arduino board.

3. Experiment System Specification :

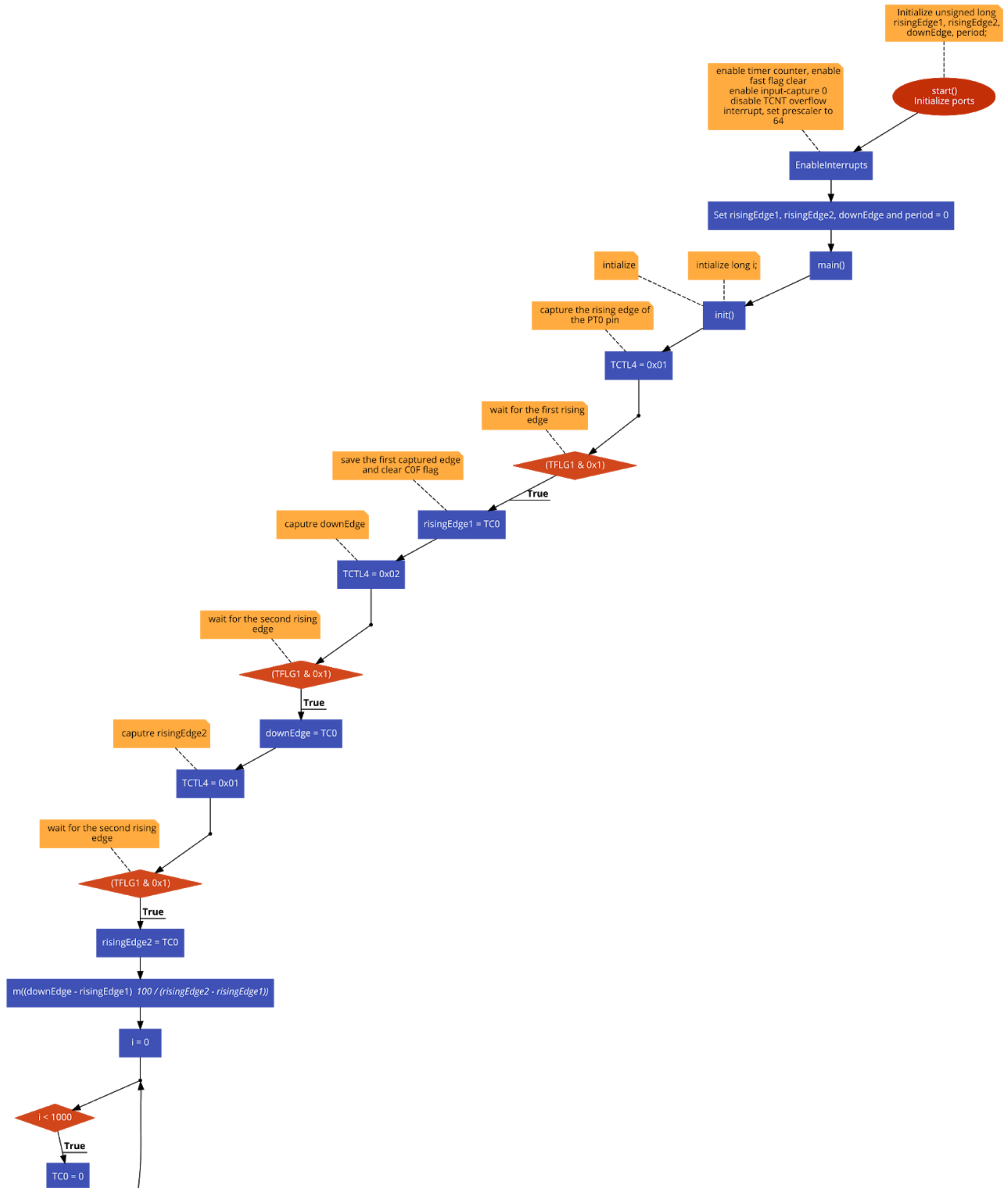
***Lab hardware photo*:**

9S12 Output: Oscilloscope:

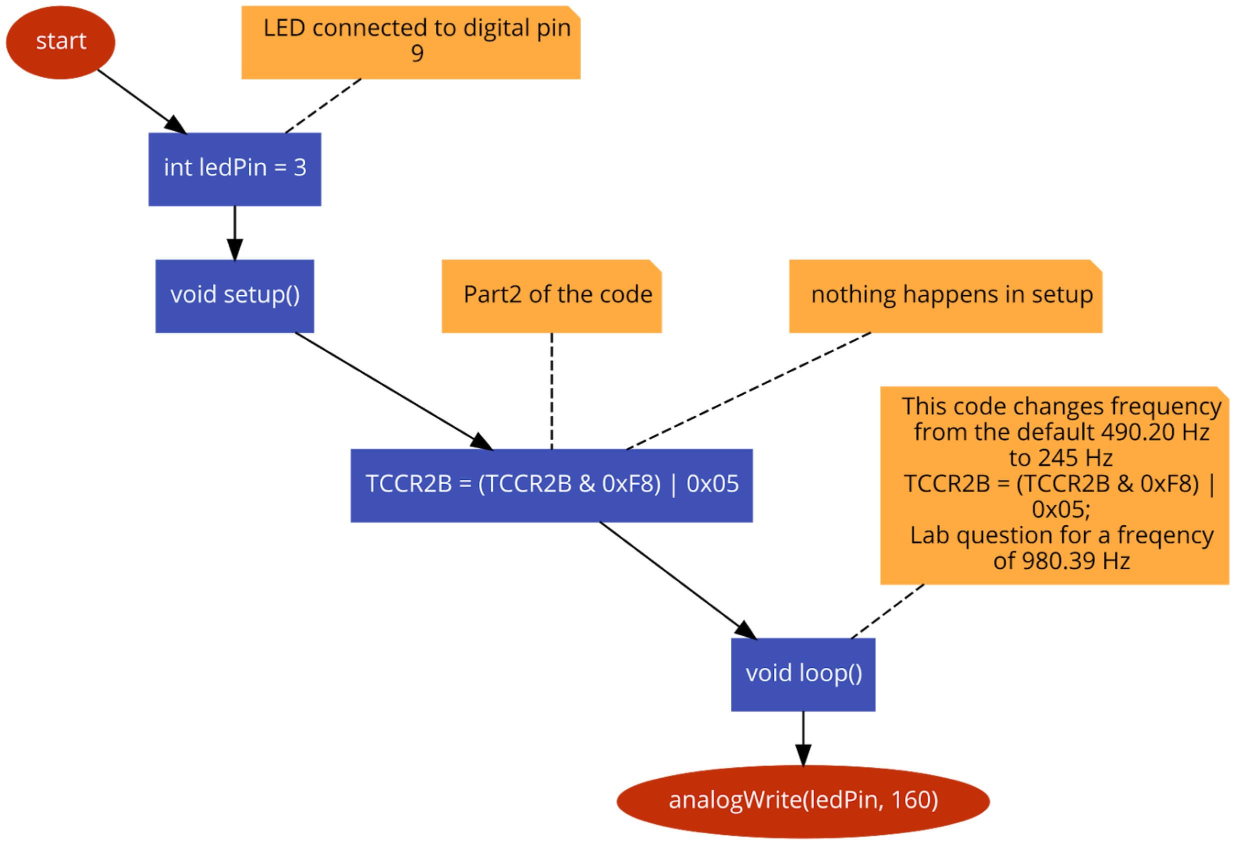




Flowchart; 9s12 Microcontroller

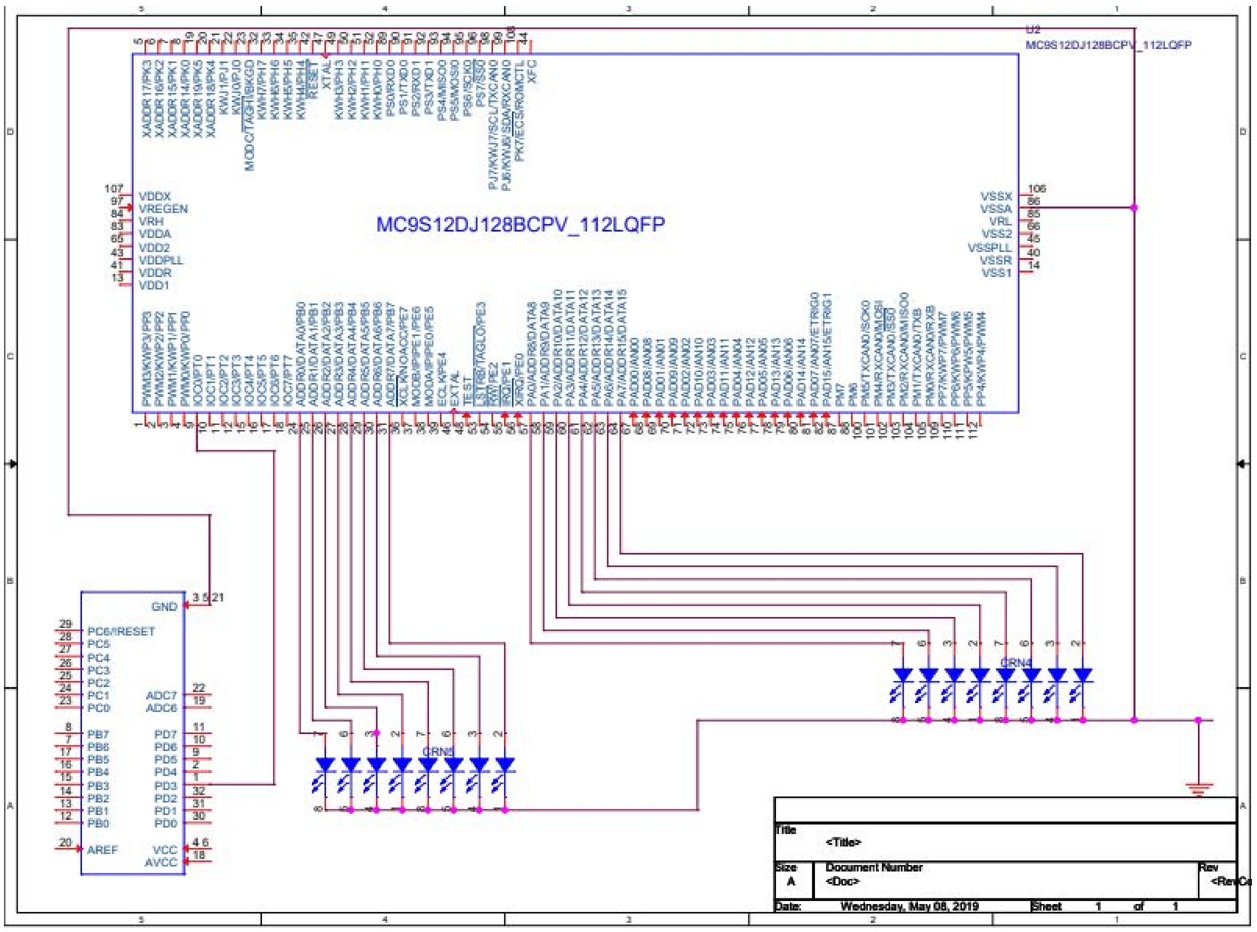

Initialize unsigned long risingEdge1, risingEdge2, downEdge, period;
start() Initialize ports
enable timer counter, enable fast flag clear enable input-capture 0 disable TCNT overflow interrupt, set prescaler to 64
EnableInterrupts
Set risingEdge1, risingEdge2, downEdge and period = 0
main()
intialize
intialize long i;
init()
capture the rising edge of the PT0 pin
TCTL4 = 0x01
wait for the first rising edge
(TFLG1 & 0x1)
True
save the first captured edge and clear C0F flag
risingEdge1 = TC0
caputre downEdge
TCTL4 = 0x02
wait for the second rising edge
(TFLG1 & 0x1)
True
downEdge = TC0
caputre risingEdge2
TCTL4 = 0x01
wait for the second rising edge
(TFLG1 & 0x1)
True
risingEdge2 = TC0
m((downEdge - risingEdge1) 100 / (risingEdge2 - risingEdge1))
i = 0
i < 1000
True
TC0 = 0

Arduino board ;


start
LED connected to digital pin 9
int ledPin = 3
void setup()
Part2 of the code
nothing happens in setup
This code changes frequency from the default 490.20 Hz to 245 Hz TCCR2B = (TCCR2B & 0xF8) | 0x05; Lab question for a freqency of 980.39 Hz
TCCR2B = (TCCR2B & 0xF8) | 0x05
void loop()
analogWrite(ledPin, 160)

4. Hardware Design :

9S12 Controller;



5. Software Design:

- ***Dragon Board Code (Codewarrior) Part1***

```
 #include <hidef.h> /* common defines and macros */
#include <mc9s12dg256.h>
#include <stdio.h>

unsigned long risingEdge1, risingEdge2, downEdge, period;

void init(){
  DDRA = 0xff;
  DDRB = 0xff;
  PORTA = 0x00;
  PORTB = 0x00;
  DDRT = 0x00;

  TSCR1 = 0x90; /* enable timer counter, enable fast flag clear*/
  TIOS &= ~0x1; /* enable input-capture 0*/
  TSCR2 = 0x03; /* disable TCNT overflow interrupt, set prescaler to 64 */

  EnableInterrupts;
  risingEdge1 = 0;
  risingEdge2 = 0;
  downEdge = 0;
  period = 0;
}

void m(unsigned long x){
  unsigned char decoder[10] =
{~0x3F,~0x06,~0x5B,~0x4F,~0x66,~0x6D,~0x7D,~0x07,~0x7F,~0x6F };
  if (x >= 100) {
    x = 99;
  }
  PORTA = decoder[x/10];
  PORTB = decoder[x%10];
  return;
}

void main(void) {
  unsigned long i;
  init();// initialize

  while (1) {

    TCTL4 = 0x01; /* capture the rising edge of the PT0 pin */
    TFLG1 = 0x1;
    while (!(TFLG1 & 0x1)); /* wait for the first rising edge */
    risingEdge1 = TC0; /* save the first captured edge and clear C0F flag */

    TCTL4 = 0x02;  // downEdge
    while (!(TFLG1 & 0x1)); /* wait for the second rising edge */
    downEdge = TC0;

    TCTL4 = 0x01;  //risingEdge2
    while (!(TFLG1 & 0x1)); /* wait for the second rising edge */
    risingEdge2 = TC0;

    m((downEdge - risingEdge1) * 100 / (risingEdge2 - risingEdge1));
    for(i = 0; i < 1000; i++){
      TC0 = 0;
    }
  }
}
```

- ***High Level Description (Part 1) Dragon Board***
  1. Initializing PortA and Port B for outputs while we will be configuring DDRT/PortT for PWM input.
  2. Enable Timer counter in such a way to also enable flag clear, and input capture to be set at “0”. Set the prescaler to 64.
  3. Enable Interrupts and initialize all the variables.
  4. Create a separate function for decoder, usually can be copied from the previous labs as we are continuing with the same ports for 7 segment displays.
  5. While in the main function, Capture the rising edge of the PT0 pin and save it in TC0 and clear the COF flag, continue with the capture of the downEdge and save it in TC0.
  6. After this, capture the rising edge for the second time and then repeat step 5 again.
  7. After storing all the values, use the formula given below to pass into the decoder function for the corresponding value output.
  8. Formula -> (downEdge - risingEdge1) * 100 / (risingEdge2 - risingEdge1)
- ***Arduino Board Code (Arduino) Part1 and Part2***

```
// LED connected to digital pin 9
int ledPin = 3;
void setup() {
  // nothing happens in setup
  TCCR2B = (TCCR2B & 0xF8) | 0x05; // Part2 of the code
  // This code changes frequency from the default 490.20 Hz to 245 Hz

  // TCCR2B = (TCCR2B & 0xF8) | 0x05;
  // Lab question for a freqency of 980.39 Hz
}

void loop() {
  analogWrite(ledPin, 160);
//  analogWrite(ledPin, 127.5);
//  analogWrite(ledPin, 252.45);
}
```

- ***High Level Description (Part 2) Arduino***
  1) Set digital pin to pin3 on Arduino Board.
  2) If it needs to be, adjust prescale values of TCCR2B with settings of prescale factor set by Arduino Board.
  3) Higher the factor value, lower the frequency.
  4) Output measured values to 7 segment LED via analogwrite with selected pin and duty cycle frequency.

6. Problems Encountered:

There were no major problems with this lab. The lab went pretty smooth, The minor problems which we did face was the fact that our arduino code was not working, but we realized that we did not need a loop for analogwrite and just use one line of code.

8.Questions:

Pre-Lab Questions:

1) **What is the notion of common ground in electronic circuits? Do Arduino and 9S12 need common ground in this lab.**

   Ans) Both digital and analog parts of the electronic circuit have their own grounds, but both of them are referenced to the same point. So, what is done is that all grounds(digital and analog) are taken together and then finally connected at one point also called "the common ground". Yes, Arduino and 9S12 need a common ground for this lab.

2) **If the Arduino is set up to do PWM at 490 Hz and the duty cycle is 30%, what is the duration that the signal is high and the duration that the signal is low for one cycle?**

   Ans) High = 490Hz * 30% = 147Hz
   Low = 490Hz * 70% = 343Hz
   Peroid is 0.002 seconds
   0.0006 seconds high
   0.0014 seconds low

Post-Lab Questions:

1.) **When the PWM circuit modulates the LED, what is the minimum frequency you can use before you start to see the LED blink?**

   Ans) Around 50Hz

2.) **For the input capture circuit, what prescale values did you choose to use? When would you make the prescale value higher or lower?**

   Ans) We used prescale of:
   32 - 980.39 Hz
   64 - 490.20 Hz
   128 - 245.10 Hz

   The settings of the prescale values are a way to measure higher and/or lower frequency.
   Higher prescale values for lower frequency, and lower prescale values for higher freqency.

1) **Provide the source code of your implementation of the analogWrite() function for HCS12 which generates a waveform with the frequency of 980 Hz**

Ans)

```
// 0x03            32        980.39
TCCR2B = (TCCR2B & 0xF8) | 0x03;
analogWrite(ledPin, 385);
// about 50% duty cycle
```

9. Conclusion:

This Lab was overall really interesting as we got to work with 9S12 more and understand about the A/D converter and also work with 7-Segment displays. This lab was not just limited to working with the 9S12 controller but also the arduino uno board and their integration, where we got to work with Arduino programming and learn more about the PWM output and input capture. For this Lab, Both Leya and Tanish worked together on software and hardware designs