EEET2481 – Final Assignment (30%)

# Exercise 1 – SPI and ADC (7 marks)

## 1.1 Requirement

Write an embedded software program for the following hardware diagram. We will use the ADC and SPI of NUC140 MCU.

Diagram

Description automatically generated

The program has the following specifications:

* Clock for the CPU is at 50 MHz
* ADC channel 7 (12-bit A/D converter with reference voltage, Vref = 5 V ) is used to continuously sample analog voltage at GPIO port A pin 7 and convert it into a corresponding digital value. The configuration for ADC:
  + ADC clock frequency is 1 MHz
  + Operating mode: continuous scan

As the program runs, as users change the potentiometer, the input voltage at GPIO port A pin 7 changes along. If the voltage is **greater than 2 V**, the program will be sending data out on the SPI interface to an off-chip slave device. In specific we will send out the 4 digits “**2022**” continuously.

* SPI2 of the NUC140 MCU will be configured as a master device to transmit data to an off-chip slave device (in this Assignment, we don’t have to worry about what device is this). The configuration for the SPI2:
  + SPI serial clock is 1 MHz
  + SPI serial clock is IDLE at HIGH state
  + Data bit is transmitted on the POSITIVE edge of serial clock
  + Data is transferred from LSB first
  + One BYTE of data to be transmitted/received in a transaction
  + SPI2 pins: GPIOD.0 (SPI2\_SS), GPIOD.1 (SPI2\_CLK), GPIOD.3 (SPI2\_MOSI)

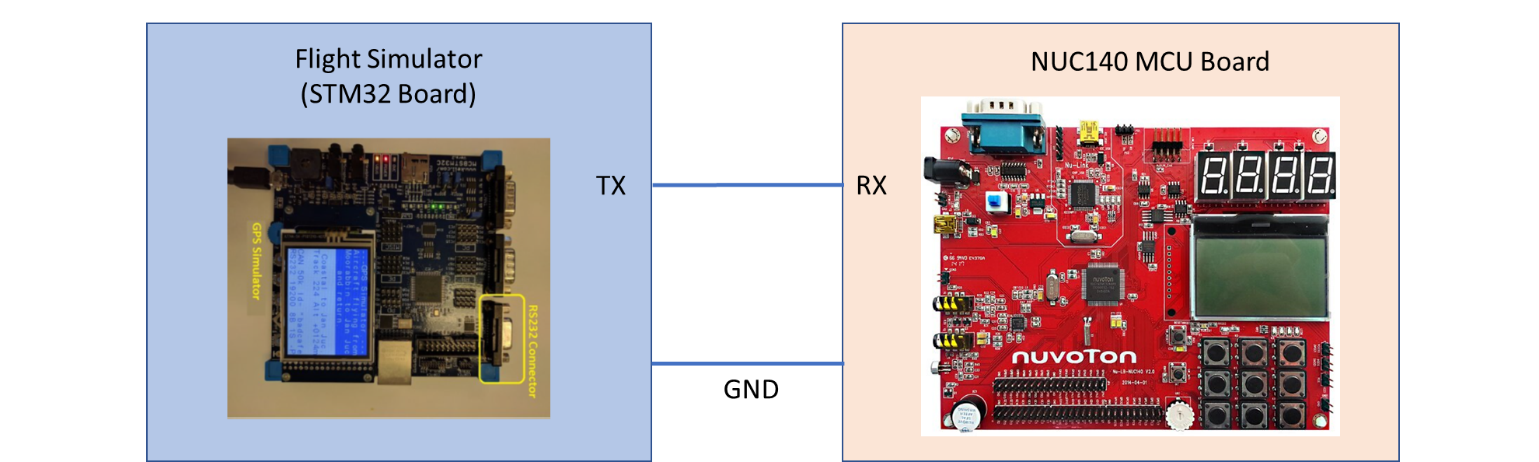
## 1.2 Submission

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. A screenshot of your waveform showing the data coming out from SPI2
3. Students will demo live the system to the Lecturer and prepared to answer questions related how they design the program.

# Exercise 2 – UART and LCD (8 marks)

## 2.1 Overview

In this exercise you will be develop a program that will run on the NUC140 board, to capture the UART data sent from another embedded system board via UART connections. The diagram of the system can be simplified as follows:



A STM32 board will act as a **flight simulator** and constantly sending data via UART connections. The flight simulator constantly sends out the flight information for a simulated flight trip. Data will be arranged in packets as shown follows:



Students will need to profile the data coming from the GPS Simulator board prior writing the program to capture and process the data. For example, by using VB or OSC, you can confirm the duration between packets, or you can see how characters are sent continuously and baud rate.

All packets follow a consistent format. An example of a packet can be seen as follows

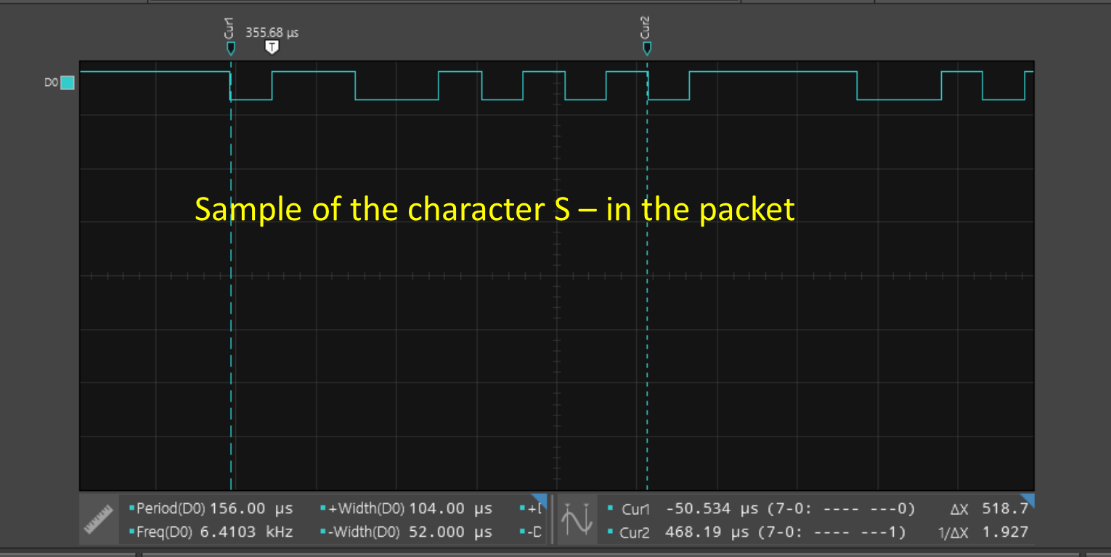
Text

Description automatically generated with medium confidence

Details for each data field:

* SONY10: Software and version. (1.0)
* 9112140152321. Date and Time. Year 1991, December 14th Sunday 15 hours (3pm) 23 minutes and 21 seconds
* S3537514: Latitude - South 35 degrees 37 minutes 51.4 seconds
* E13944402: Longitude - East 139 degrees 44 minutes 40.2 seconds
* +2365: Altitude Meters
* 234: Speed km/hr.
* 271: Bearing/track (made good) in degrees relative to True North
* 9112140152320: Date and time the data acquired. If (actual) time given earlier is more than 1 second greater than the GPS is using old data.
* A: Dilution of precision. A DOP = 1, B = 2 etc. Q 100+
* 3: 3 Satellites used: (2D mode) A: WGS-84 map datum used.
* RHECT etc.: Satellite Information. R = satellite 18, H its between 65 and 75 deg elevation. E 36 to 45 degrees east of true north, C Satellite healthy and can be used (F used). T signal strength high (A low through Z high).

If you profile the data and focus on each character you can observe the UART data on VB or OSC:



As the simulator simulate an actual flight, you should expect the data field of the

## 2.2 Requirements

In this exercise your program supposes to capture the data sending from the Flight Simulator board and extract only the Latitude and Longitude from all the packets and then constantly show these on the LCD of the NUC140. We should expect these data fields will be update occasionally, depending on the actual data sending from the simulator.

Graphical user interface

Description automatically generated

## 2.3 Submission

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. Some screenshots to show you profile a sample packet. You need to
   1. Extract one packet and identify the start of the packet
   2. Identify the baud rate of the transmission
   3. Take 3 samples characters and show their equivalent waveforms on OSC or VB.
3. Students will demo live the system to the Lecturer and prepared to answer questions related how they design the program

# Exercise 3 – Battleship Game (15 marks)

In this exercise you will design a simple battleship Game using NUC140 Board. If you are unfamiliar with the game, you can check out some videos on YouTube on how to play it. We will only implement the single-player mode. The game field is loaded in advance, and the player will play until the game ends. There will be indications if there are misses, hits and game overs. Technically, you are required to work with UART communication, GPIO, Timers and Interrupts.

## 3.1 Game Field

The game field is to be 8x8 locations. For example, the following game field has **five** ships (each ship consists of 2 dots). For example:

- - - - - - - -

- X X - - - - -

- - - - - - X -

- - - - - - X -

- - X X - - - -

- - - - - - - -

X X - - X - - -

- - - - X - - -

To create the field, we will use Notepad to create a text file with the number 0 (represent water) and the number 1 (represent part of a ship). The text file version of the above looks as:

0 0 0 0 0 0 0 0

0 1 1 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 0 0 1 0

0 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0

1 1 0 0 1 0 0 0

0 0 0 0 1 0 0 0

Two sample fields will be provided for your information. You can also create other files using Notepad if required.

## 3.2 System Description

1. After reset, the LCD shows **a welcome screen**. You can flexibly to choose the texts and images to show on this screen.
2. Then, if you send a map file from your laptop/PC to the Arduino board. You need to write an embedded program to capture and store this map temporarily. You only need to do this one time before the game starts. Once the sent complete, the screen should display “**Map Loaded Successfully**” right in the middle of the screen.
3. If you press **SW\_INT (GP15)** the game starts. We will use External Interrupt in this system.



The LCD now will show the full map where - representing water

- - - - - - - -

- - - - - - - -

- - - - - - - -

- - - - - - - -

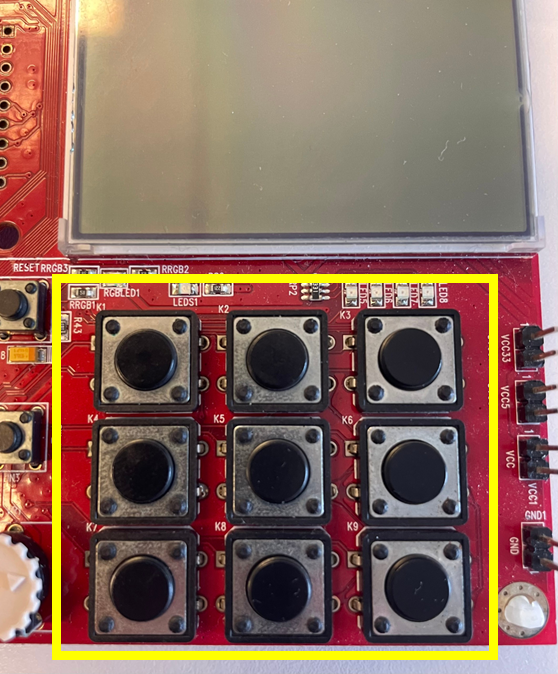
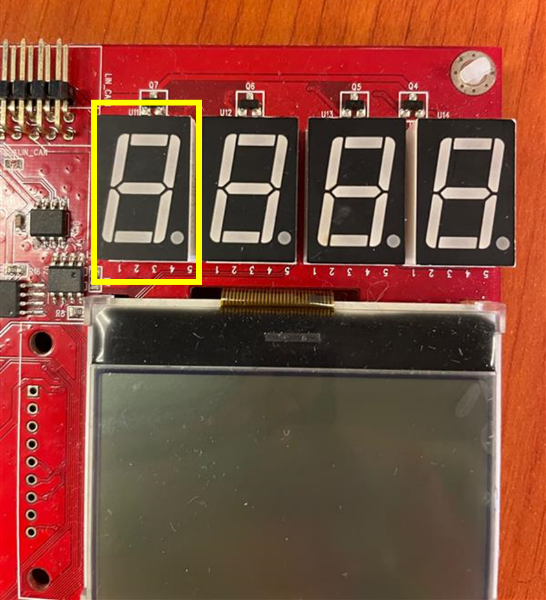
- - - - - - - -

- - - - - - - -

- - - - - - - -

- - - - - - - -

1. Select **coordinate X** (ranging from 1 to 8) by using press the corresponding Key 1 to Key 8 from the Key matrix. This current coordinate X also displays on the 7segment U11.

1. Press Key 9 will allow us to change coordinate X
2. Repeat step d to select **coordinate Y.** This current coordinate Y also displays on the 7segment U11. If you press the Key 9 button again, it will change back to coordinate X selection.
3. Once XY is defined, we can press the **SW\_INT (GP15)** to shoot

A picture containing text, electronics, circuit

Description automatically generated

Your program then compares the entered coordinates vs the map earlier and then indicates if there is a Hit or Miss. If there is a **Hit**, we will flash **the LED5 (GPC12)** 3 times three times. You can choose the suitable flashing frequency here.

If there is a hit, the LCD updates to indicate the hit. For example, X display the hit:

- - - - - - - -

- - X - - - - -

- - - - - - - -

- - - - - - - -

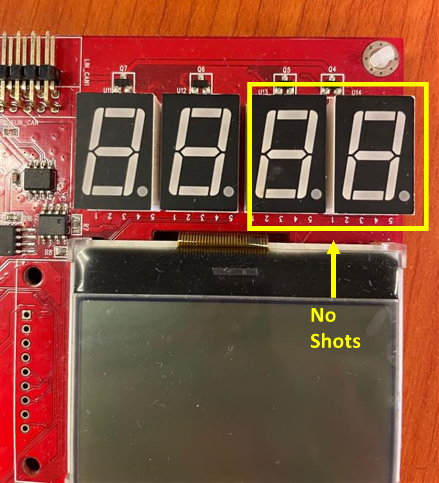
- - - - - - - -

- - - - - - - -

- - - - - - - -

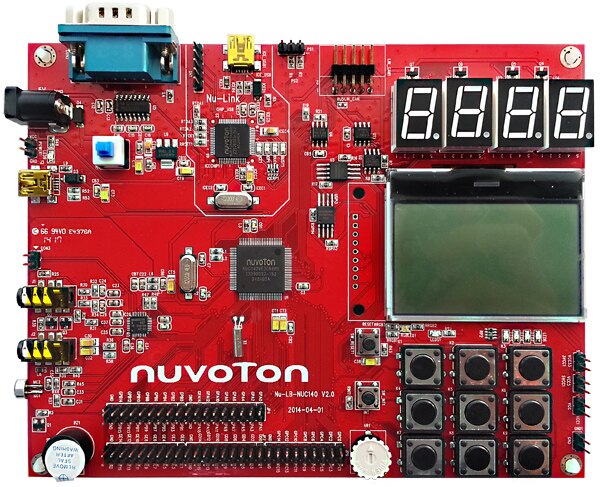
- - - - - - - -

1. The system also displays in real-time the number of shots fired via U13 and U14:



To control this compound 7segment-LED, we will need to apply the scanning technique to select and display one digit at a time, then change it to the next digit. The scanning rate must be quick enough, so all the numbers appear visible to our eyes. You must use a **Timer interrupt** to control the scanning rate.

1. If the player successfully sinks all five ships on a map (Note that a ship is sunk if two hits are at two adjacent dots) or has fired more than 16 shots but could not sink all ships, the game will be over. Couple indications for this:
   * + - The LCD will display Game over message.
       - Buzzer will go off 5 times one time only.



* + - * All the keys will stop responding.

1. If you press the **SW\_INT (GP15)** now, the game will be replay from the welcome screen

## 3.3 Other Requirements

The system must include the following game features:

* Check if a ship is shrunk when two adjacent dots are hit.
* Indicate if the same position is hit twice. For this, the number of shots will go up by one still.
* Count maximum shots allowed for a player; if he runs out of **16 shots**, then the game is over.

In terms of technical requirements, your system must use:

* Program is developed using **registered based method.** All configurations must be clearly explained (via code comments). Only standard drivers for LCD display (i.e., functions in LCD.c and Draw2D.c libraries) can be used in project code.
* Design with FSM
* UART Interrupt to receive text file sent from PC. You can decide the baud rate and UART format.
* Timer Interrupt for scanning rate. You can use this to define the delay time between the display of each digit.
* External Interrupt for **SW\_INT (GP15)** button
* Use C functions to optimize the code.
* Put meaningful and clean comments to explain your design
* The entire system is built properly on a breadboard and ready to be checked.

## 3.4 Submission

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. Other supporting documents: State diagram , and other documents that help to explain your design understanding.

## 3.5 Demonstration

A project demonstration to present how the system works in practice and Q&A with the lecturer on project-related matters.

In addition, you might also be asked one more open question regarding what we have learned in the class, including GPIOs, Timers, Interrupts, SPI, LCD, ADC and UART.