Sabotage

Individual Report

Moiz Lokhandwala

27669837

Team 11

Table of Content

GPS Module

I worked on writing the GPS driver in C which was used to receive GPS data from our GPS module. This data is critical as it is used to display the De1-Soc location on the app, and finding the De1-Soc is the whole premise of our game.

To communicate with the module through RX and TX serial communication. I used the altera_avalon_uart IP. I use the functions getc() and putc() to receive or send characters through serial respectively. The settings I used for the IP are given below.

- Parity: NONE
- Data bits: 8
- Stop bits: 1
- Baud rate: 9600

Hardware Setup

The TX and RX are linked to two GPIO pins on the De1-Soc. I then connect the RX to the TX on the module and vice versa. I also have to connect the 3.3v pin and GND to the GPS module.

There are five different types of output messages transmitted from the module to the De1-Soc. These include GGA, GSA, GSV, RMC and VTG.

The RMC (Recommended Minimum Navigation Information) is of interest to us as it provides the position data such as longitude, E/W, latitude and N/S. This data is in NMEA and would need to be parsed to get the relevant information.

Example of an NMEA formatted message $OutputSentence,OutputValues*Checksum

Implementation

Every output message starts with a dollar sign, therefore I developed a function read_serial that takes in the type of message (RMC, GGA etc) as a parameter and it would wait until it receives a message of that type. It would then fill up a raw buffer with the characters from the message.

The parse_raw_buffer function then parses the raw buffer with the comma-separated section and populates an ouput_buffer. When position data is unavailable the longitude and latitude section in the message is empty. I needed to send '0' to the server in the case when we cannot receive valid data to indicate that no new position data is available. This is not possible when using strtok from the standard library as it would not detect the empty section in the message (When data is not available). Therefore I needed to write a parsing function that would add the '0' character to the appropriate index of the parsed buffer array.

The GPS data is sent to the server through the WiFi module periodically. This is achieved through a timer interrupt where a ready flag would be set in its ISR. The flag is then cleared once receiving GPS data and sending it to the server.

The longitude and latitude values from the module were in degrees decimal minutes (ddmm.mmmm) while the app needed values in degrees decimal. I wrote a function to convert these values using the formula:
Conversion to decimal degrees = degrees ('dd') + minutes/60 ('mm.mmmm')

Example GPS Data

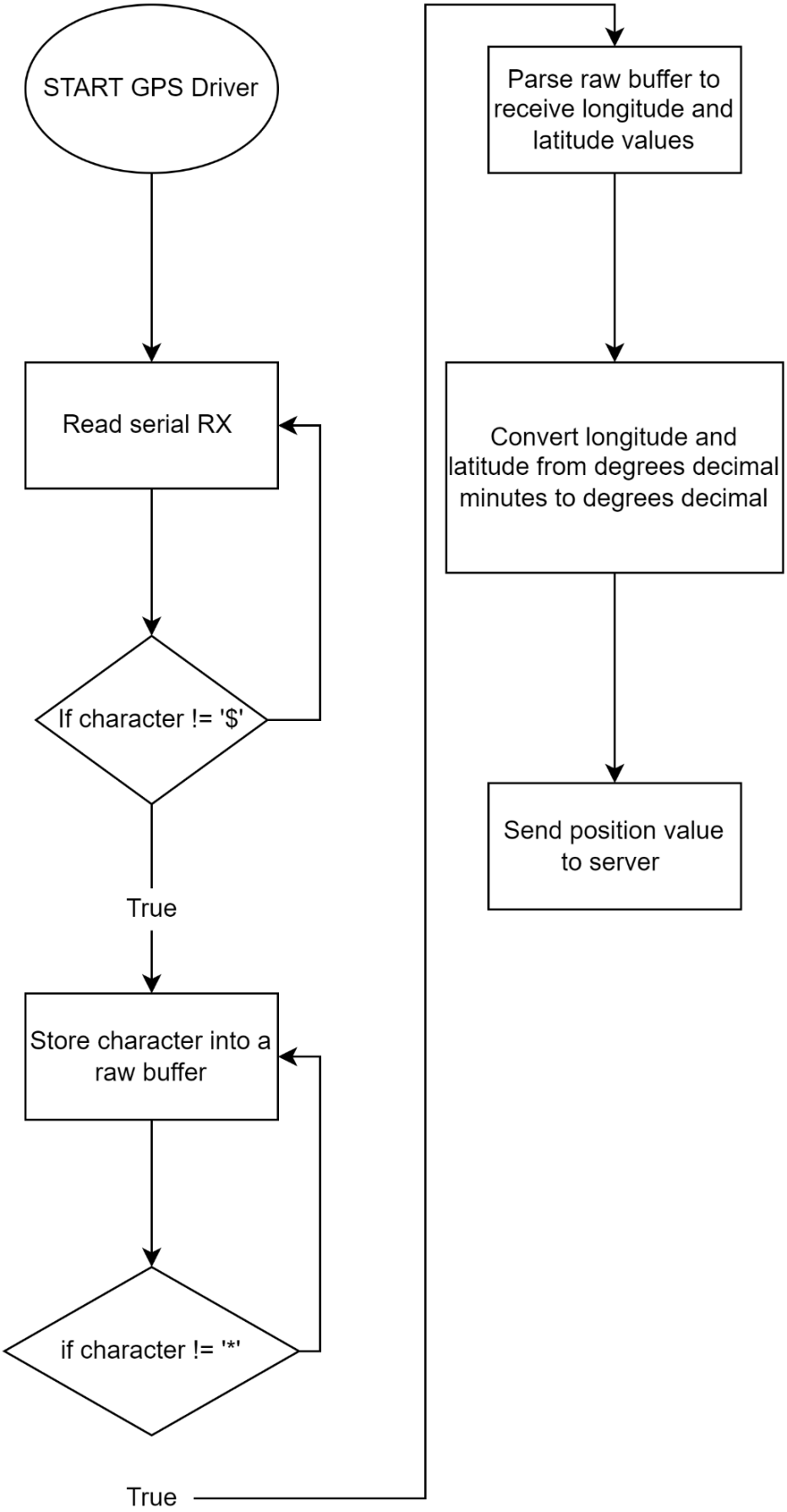
Cloudy Weather [11th March]

Under the bed (Invalid data)

- GPRMC,010832.000,V,,,,,0.59,59.77,120322,,,N
- GPGSA,A,1,,,,,,,,,,,,,,,
- GPVTG,75.29,T,,M,0.36,N,0.67,K,N
- GPGGA,010833.000,,,,,0,03,,,M,,M,,
- GPGSV,3,1,10,02,73,079,,12,67,161,,25,64,286,30,20,37,133,

Window (Valid data)

- GPRMC,011921.000,A,4916.1724,N,12314.9628,W,0.13,351.37,120322,,,A
- GPGSA,A,2,25,31,29,,,,,,,,,,4.77,4.66,1.00
- GPVTG,141.05,T,,M,0.08,N,0.14,K,A
- GPGGA,011922.000,4916.1725,N,12314.9628,W,1,03,4.66,15.8,M,-16.9,M,,
- GPGSV,2,2,08,29,39,278,33,06,23,051,16,05,22,153,,31,20,313,22


START GPS Driver
Read serial RX
If character != '$'
True
Store character into a raw buffer
if character != '*'
True
Parse raw buffer to receive longitude and latitude values
Convert longitude and latitude from degrees decimal minutes to degrees decimal
Send position value to server

VGA Touch screen

We used the touch screen to display a number pad. the player who finds the De1-Soc would use this to type in their user ID to register their win on the server. I use the altera_up_avalon_video_dma_controller to output characters and pixels on the screen.

Implementation

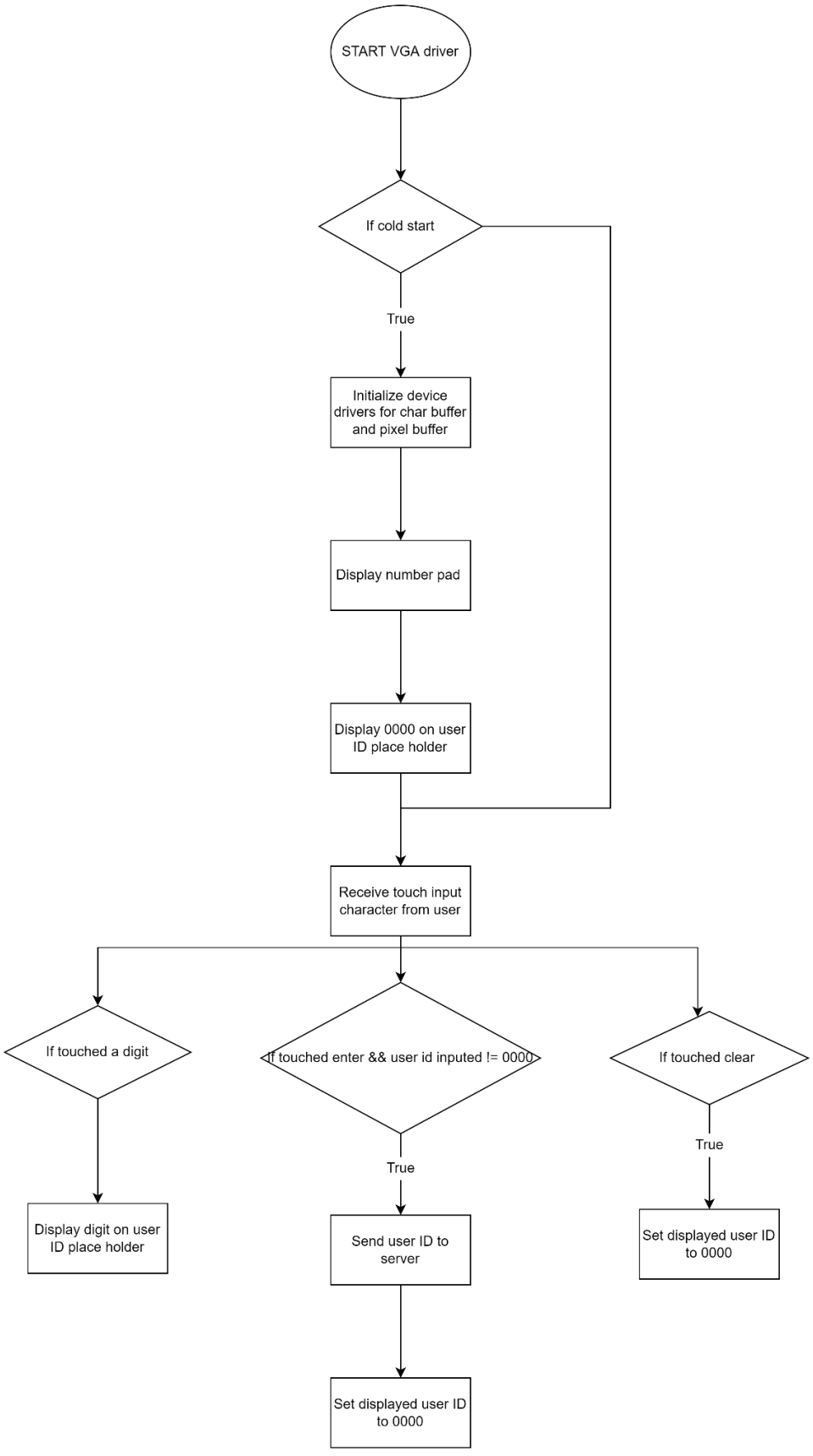
The draw_numpad function only runs once when there is a cold start and draws the number pad on the screen.
Format of the number pad:

7 - 8 - 9
4 - 5 - 6
1 - 2 - 3
0 - en- cl

On the VGA pixel subsystem buffer, the squares of the number pad start at (40, 280) with a size of 40 on each side. The character subsystem buffer has a different coordinate system and instead starts at (15, 75) which would be at the center of the square box and has a size 10 gap between each character.

The VGA driver received input of characters touched by the player from the touch screen driver. The is then passed through an input handler function which decides what to do when a certain character is selected.

- 0-9: would call the add_digit function that displays this character on the screen and adds it to a user_id array for tracking purposes
- Enter: only when the user ID is not “0000” (which is the default value) it would sent an external variable used in the main function to the current user ID. The main function would then send this user ID to the server
- Clear: calls the function clear_user_id which sets each character on the digit placeholder to zero


START VGA driver
If cold start
True
Initialize device drivers for char buffer and pixel buffer
Display number pad
Display 0000 on user ID place holder
Receive touch input character from user
If touched a digit
If touched enter && user id inputed != 0000
If touched clear
True
True
Display digit on user ID place holder
Send user ID to server
Set displayed user ID to 0000
Set displayed user ID to 0000

Bluetooth Module (Discontinued)

Our initial intention was to utilize the Bluetooth module to gather RSSI data from nearby players' phones. This would then be used to calculate proximity and allow for the hot and cold functionality. After various testing, I found out that the range of Bluetooth was too short. We decided to use RSSI from WiFi RSSI through the app instead as it covered a larger distance and was more reliable.

Implementation

Default config for Bluetooth module

1. Bluetooth slave mode
2. Bluetooth pin code 1234
3. Baud rate - 115,200 kbps, 8 bits, no parity, 1 stop bit

To configure the module we need to use the RS-232 DB9 port connected via USB cable to the computer. You then need to enter command mode by opening Putty and typing in '$$$'. The module would then return 'CMD'. The command mode is used to configure the Bluetooth device or receive RSSI data from nearby Bluetooth-enabled devices. I intended to send it the command 'IQ' which scans for devices and returns their RSSI. This article (https://medium.com/beingcoders/convert-rssi-value-of-the-ble-bluetooth-low-energy-beacons-to-meters-63259f307283) contains a formula that can be used to convert RSSI values into meters to get an accurate measurement of the distance from the De1-Soc.

States of the Bluetooth device when trying to connect

1. Discovery - The Bluetooth module broadcasts its name profile support and MAC address (ready to be paired with other devices)
2. Pairing - Form a secure pairing with the device. Exchanges security keys
3. Connecting - Must be paired successfully before connecting

Team Effective

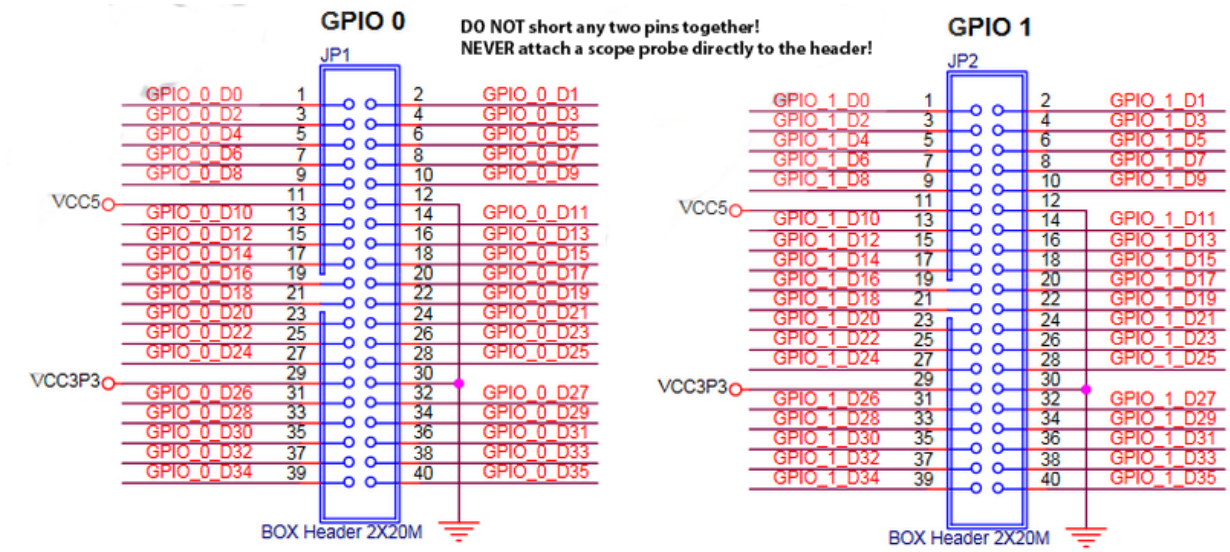
Our team worked efficiently throughout the project as everyone had their own tasks to work on. We were also able to collaborate well through Github, but there was some disorganization related with merging branches and our workflow.

Challenges

The data received from the GPS wasn't always accurate and would have a difficult time finding a satellite in view. It would also not be able to receive valid longitude and latitude data in certain buildings. In future projects, I would prefer if we use a portable power supply, so the De1-Soc can be placed outdoor.

Another challenge I faced was working on the Bluetooth module as we did not plan on using it in our final product. This wasted time that I could have spent on other productive aspects of the project.

Pin Assignments



GPIO_0_D34 - GPS_RX

GPIO_0_D35 - GPS_TX

GPIO_0_D32 - Bluetooth_RX

GPIO_0_D33 - Bluetooth_TX

VCC3P3 - GPS_VIN

VCC3P3 - Bluetooth_VIN