Individual Report - Bowei Ren

Individual Identification

Name: Bowei Ren

Student Number: 42734889

Team Number: L2B-11

Project Title: Sabotage

Individual Contribution

Sprint 1:

- build a framework for touch driver, including receiving and sending packets via UART connection (work alone)

Sprint 2:

- The touch screen driver is fully functional (work alone)
    - now able to send commands/receive reports with AR1100 touch controller
    - record every touchpoint with a pen down/up (press & release) information

Sprint 3:

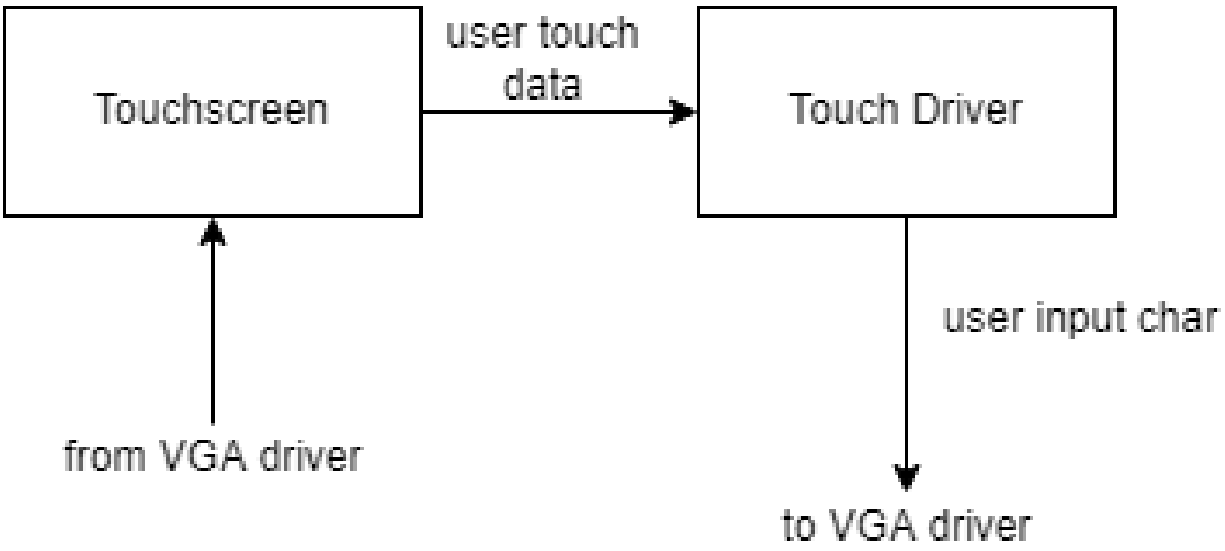
- Integrate with the VGA driver so that the player's input can be processed and sent to the wifi module (work with Moiz)

Final Demo:

- Integrate all hardware components including GPS, WiFi, touchscreen, and VGA (work with Moiz)

Detailed Design

- **Block Diagram of my Hardware Part (individual work)**



- The driver is running on the NIOS system, and it is written in C.
- The driver is designed to communicate with AR1100 Resistive Touch Screen Controller under GENERIC mode.

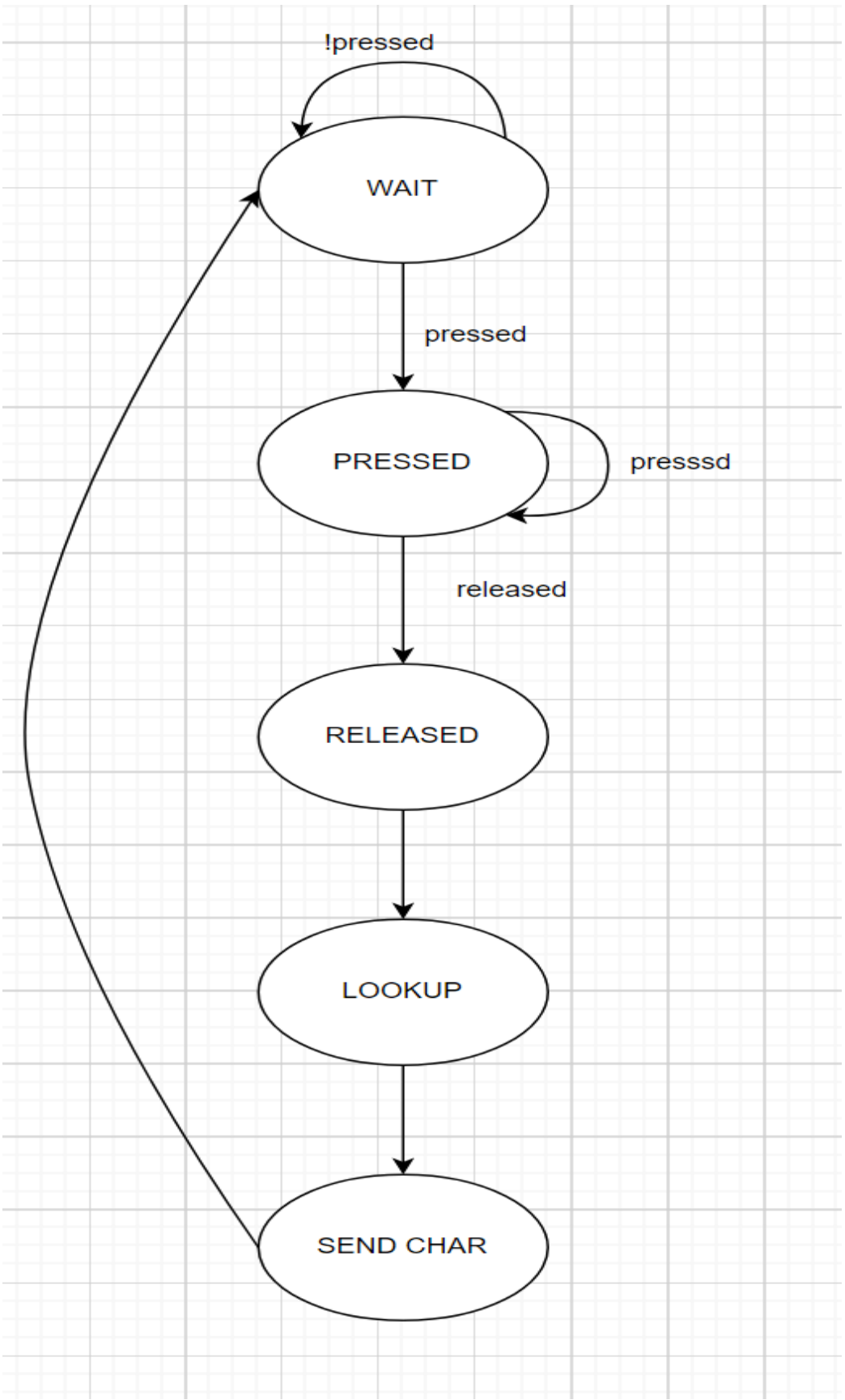
- **Major Data Structure**

```c
/* a data type to hold a point/coord */
typedef struct { int x, y; } Point ;

// structure for command and response packets
boweiren, 3 weeks ago | 1 author (boweiren)
typedef struct {
    unsigned char sync;
    unsigned char size;
    unsigned char cmd;
    unsigned char data[MAX_INTERRUPT_OUT_TRANSFER_SIZE-3];
} Command;

boweiren, 3 weeks ago | 1 author (boweiren)
typedef struct {
    unsigned char sync;
    unsigned char size;
    unsigned char status;
    unsigned char cmd;
    unsigned char data[MAX_INTERRUPT_IN_TRANSFER_SIZE-4];
} Response;
```

- **Touchscreen Driver Control Flow**



- **Summary**

    - In GENERIC mode, each controller's response contains 5 bytes of data. The first byte only contains pen state information (pen up / pen down). The second and third bytes contain the x-axis information of the touchpoint. The fourth and fifth bytes contain the y-axis information of the touchpoint.
    - Since resistive touch screens don't support multiple touchpoints, standard user input should contain a series of press responses and one release response.
    - According to the above logic, I use the release response to indicate the end of a touch.
    - Therefore, in a complete run, the driver will initially be waiting for a response. Once it received a release response, the point will be looked up in a hard-coded lookup table to find the key. Finally, the driver is going to our put this key as a char variable.

- **Extendable Design**

    1. The touch drive is calibrated with relative coordinates. So, if we use another touchscreen with a different size, it can be easily recalibrated by adjusting four parameters.

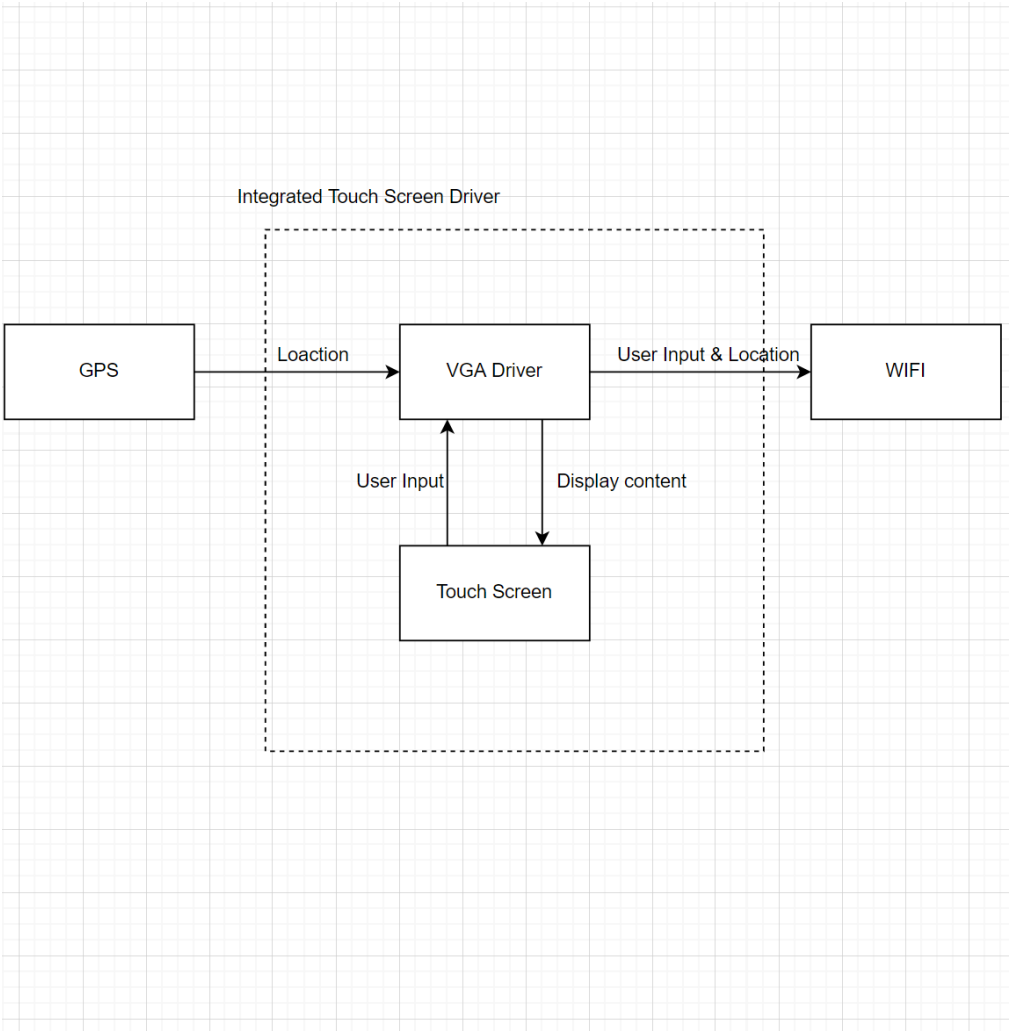
    ```
    #define KEYPAD_TOPLEFT_X        354
    #define KEYPAD_TOPLEFT_Y        583
    #define KEYPAD_BOTTOMRIGHT_X    1268
    #define KEYPAD_BOTTOMRIGHT_Y    912
    ```

    2. The driver is highly modularized, making maintenance easier.

- **Hardware Component Overview (in Shared Appendix)**

    - Refer to Shared Appendix Figure B.

- **Hardware Integration (with Moiz)**



- Firstly, we integrate the VGA driver with the touch driver to make a whole touchscreen driver, which can display a keypad on screen. It can also interact with user's touch input to display the user input number on the left side of the screen. To achieve this, we call the getTouchChar() function inside the VGA driver.
- Secondly, we integrate GPS and WiFi modules into our touchscreen driver. But here comes conflicts. Detailed descriptions can be found in the third point in the Challenges part below.
- At last, we did some test on the overall system. We visited several spots on campus to find a place with an outdoor socket or a socket close enough to an outdoor area since the GPS module requires a clear view of sky. Eventually, we did one test on the roof of the Nest Building and another inside the EDC Building and fixed some bugs of inconsistent data structure.

Challenges

1. **Problem:** When testing my driver, I notice the touch screen is not properly calibrated.
   Solution: I did some experiments and add those inaccuracies into the driver software. And I also use relative coordinates instead of absolute coordinates to minimize errors.
2. **Problem:** There is some noise in touch data.
   Solution: I first try to implement some touch algorithms from a research paper (Mohamed, 2014). But the algorithm takes too much time and make the touch screen not that responsive. Eventually, I notice that pressing the keypad is different from other complicate touches, such as swiping or drawing lines. I can easily use the release point as the touchpoint. So the final driver only takes
3. **Problem:** The touch drive has to busy-wait for user's touch input, but the GPS driver is supposed to send location data periodically. Our program is single-threaded, so we must avoid busy waiting.
   Solution: So we rewrote the getc() function from stdio.h library to add a timeout to the getTouchChar() function. Also, we added interrupt handlers for periodic GPS location report.

Team Effectiveness

Generally speaking, our group works efficiently. However, the problem of lack of communication still persists. Meanwhile, team members rarely push their latest code to GitHub but communicated their progress verbally instead. This resulted in members not having a deeper understanding of other people's tasks.

Other Comments

- How did you ensure that your tasks would integrate with your team?
  We communicate with other team members frequently during development. Therefore, we not only have a clear understanding of our own project, but also have a general concept of the whole project.

- What did you do to ensure success, or at least improve the likelihood of success?
  I believe that starting to plan your mission early is key to ensuring success. Setting goals give you a clearer idea of the progress of the task.

References

Mohamed, M. G., Jang, U., Seo, I., Kim, H. W., & Cho, T.-W. (2014). Efficient algorithm for accurate touch detection of large touch screen panels. *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*. https://doi.org/10.1109/isce.2014.6884376