ECS 193B: Team 7 AutoTracktors
Mobile App Development Manual V1

Table of Contents

About the App

Introduction

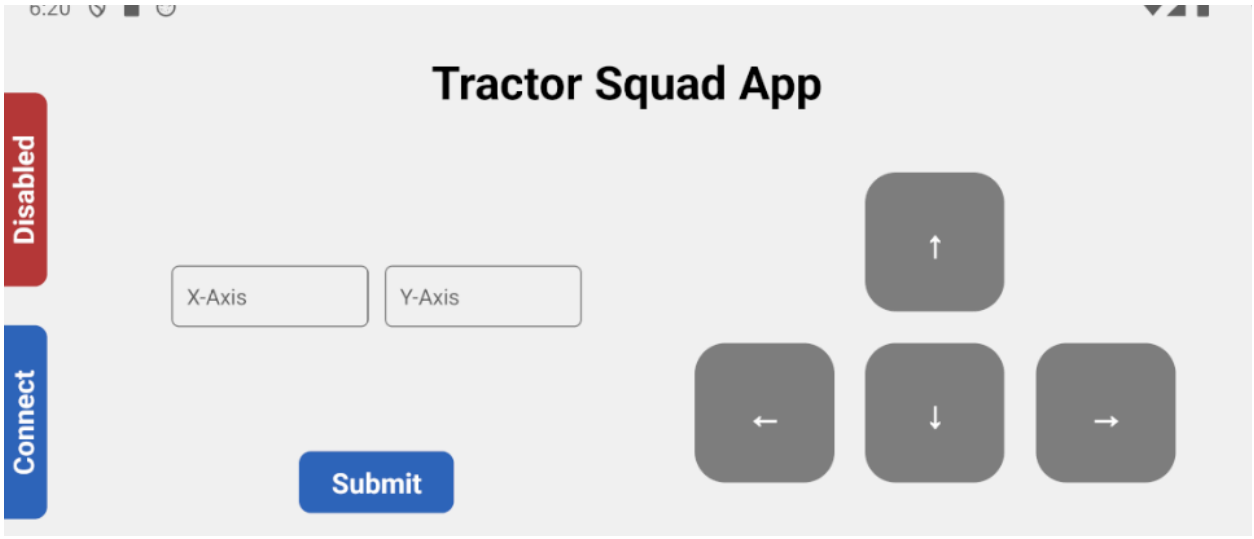
The purpose of this app is to communicate with the robotic platform and allow users to override its GPS pathing with manual controls. The reason/importance of this is in case there are timing issues with the autonomous tractor that it's being used to test with. It also allows researchers a degree of freedom and flexibility in case they want to rely solely on manual controls as well.

In addition, we want researchers to be able to input GPS coordinates from our app as we believe this will be the most convenient way (instead of the alternative of manually navigating through the Raspberry Pi and its directories)

An important note is that this app currently does not support IOS. In order to further develop this project, you will need an Android phone.

Description of the app

The current state of the app is the following:



- **Disabled/Enabled:** Upon pressing, a prompt will appear to ask users if they are sure they want to send movement signals to the Pi. Upon enabling, the red 'Disabled' will turn into a green 'Enabled'. Pressing this green enabled again won't output a prompt and instead go into disabled. Think of this as a safety, as users can accidentally press an arrow key when unintended and collide with something on accident.

- **Connect:** Will scan for bluetooth devices around the cellular device. Currently, it sees everything, however this can be easily changed to only look for the Raspberry Pi (can look for the Pi's specific mac address or device ID)
- **X axis/Y-Axis containers and Submit:** Whatever user's input into these boxes, the Pi will receive if the user presses submit. As of now, nothing stops users from putting any strings, so limiting this to some integer or rounded float is ideal.
- **Arrow Keys**: Upon pressing, a movement signal is sent to the Pi. This movement signal gets processed until finally moving a corresponding motor.

Reasons for Bluetooth Low Energy (BLE)

Because these autonomous tractors will be used in farm fields, network availability may be compromised. BLE provides a reliable way to ensure a way to communicate with the robotic platform at all times in farm fields.

Below are some studies justifying the use of BLE:

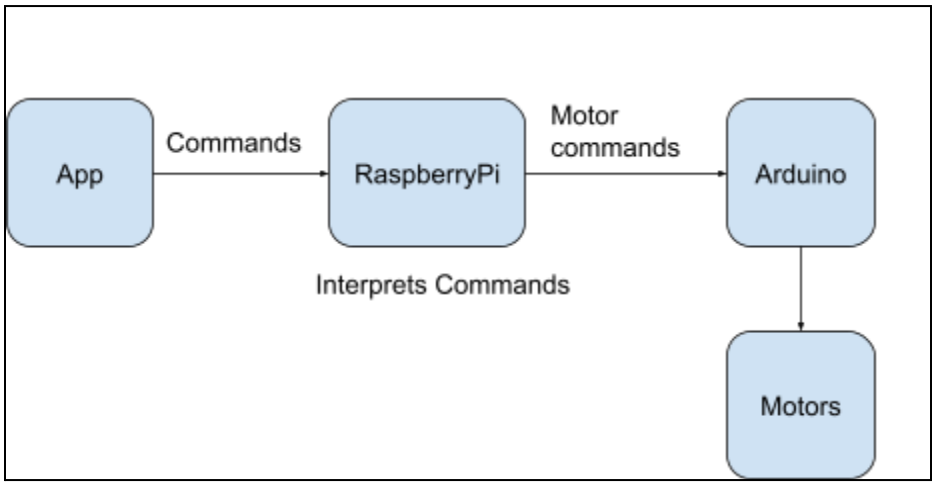
- Paper showing a wireless safety network infrastructure using BLE (bluetooth) in agricultural environments (https://www.mdpi.com/2071-1050/14/15/9393)
- Study showing the conjoined use of GPS and bluetooth for analyzing tillage paths (https://iopscience.iop.org/article/10.1088/1755-1315/355/1/012014/pdf)
- Study showing some limitations of using GPS (noise) and justification for requiring a bluetooth controller (https://link.springer.com/chapter/10.1007/978-3-319-07488-7_29)

Technologies Used

- Javascript
- React Native Expo
- Android Studio
- Visual Studio (although any IDE that supports Java should work. Visual Studio is more generic so we recommend that)

Code

Technical Details



Every arrow key, upon pressing, sends a corresponding character to the Pi. The Pi then interprets this character through code and then turns on corresponding motors. These signals/characters will be discussed below this section.

In the background of this app being run an **acknowledgement signal** is sent periodically to the Pi. This is currently configured to send one every 5 seconds. This is to ensure a consistent connection between the cellular device and robotic platform, and to abort/stop the robotic platform if the manual control state is no longer receiving this signal.

Different Signals Sent

We sent different signals from the phone app to our raspberry pi. These signals are what we will be using to notify the pi and robot of what state it should be in. Here is a list of all states and how they are interpreted from the app to the pi:

1. **Forward State:** Moves the bot forward. Notified as “u” from the app. The pi interprets this as an ascii decimal value 117.
2. **Reverse State:** Moves the bot backwards. Notified as “d” from the app. The pi interprets this as an ascii decimal value 100.
3. **Left State:** Pivots the bot left. Notified as “l” from the app. The pi interprets this as an ascii decimal value 108.
4. **Right State:** Pivots the bot right. Notified as “r” from the app. The pi interprets this as an ascii decimal value 114.

5. **Stop State:** Stops the bot but keeps it on. Notified as “s” from the app. The pi interprets this as an ascii decimal value 115.
6. **Acknowledgement State:** Tells the bot that the phone and pi are connected. Notified as “@” from the app. The pi interprets this as an ascii decimal value 64.
7. **Enable State:** Enables the robot to listen in for further move states. Notified as “&” from the app. The pi interprets this as an ascii decimal value 38.
8. **Disable State:** Disables the bot and prevents it from receiving any more move states until re-enabled. Notified as “#” from the app. The pi interprets this as an ascii decimal value 35.
9. **GPS State:** Moves the bot on a predefined route. Notified as “g” from the app. The pi interprets this as an ascii decimal value 103.

Github

https://github.com/RaphaelT1010/ecs193app

Resources

- https://www.youtube.com/watch?v=EUWn1aPATko Example project
- https://github.com/dotintent/react-native-ble-plx React Native BLE PLX Github
- https://dotintent.github.io/react-native-ble-plx/ React Native BLE PLX Documentation

We were able to develop our application with the above as the minimum. ChatGPT was also consulted for GUI matters, so we recommend we use that as well.

Limitations and Future Development

Because this is our first time doing mobile app development, many optimizations are available for the application. There are also some noticeable limitations. Here are a few:

Limitations:

- Signals can be lost. As of now, there’s no way to ensure that a signal is sent. We really just rely on the constant acknowledgements that the Pi receives. There’s also no way to resend a lost signal that is handled by code, so users must repress an input if a signal is somehow lost.

Future Development:

- Making the application available on IOS
- Have the Pi send another acknowledgement back when receiving an acknowledgement (an acknowledgement for an acknowledgement) This way, the app can know if it must resend a signal under the hood, rather than the user needing to press the same button twice in a row (this also has safety implications)