Lab 5: Bluetooth Control and Distance Mapping

Irina Tolkova # 1240609
Samuel Scherer # 1435365
Tremaine Ng # 1433161

Overview

In this lab we used a BlueSMiRF bluetooth modem from Sparkfun to enable 2-way wireless communication from our Beaglebone to any compatible device. We used this communication to visualize data on a matlab plot of the distances from the car to its surroundings in a circle. The intention was to allow an operator to generally understand the surroundings with this information and navigate through a hole in a box-shaped enclosure using bluetooth control of the car.

Hardware

Bluetooth Module

The BlueSMiRF bluetooth module was powered by +3.3V digital outputs from the beaglebone and the TX and RX pins connected to the RX and TX on UART 4 on the beaglebone board. The TX and RX were connected to the opposite module, i.e. TX to RX and RX to TX. This connects data receiving lines to data transmission lines.

To begin using the module, a bluetooth device like a smartphone or computer had to be connected via bluetooth pairing. The device had a default PIN “1234”, and once connected the user could use a program (matlab, arduino serial terminal) to send data to the module and configure or use it.

In order to write to the bluetooth module to send it to the connected computer, a terminal file ttyO<number> would be used, the number corresponding to the UART being used. This could be found by using the dmesg command in bash and seeing which terminal file was enabled.

H-Bridge

As in previous labs, the h-bridge served as a way to control the motor’s external power supply using the board’s output pins. The h-bridge’s logic inputs set the motion to one of 6 modes: Free, forward, reverse, brake, left, and right. The pin outputs were given by setting the GPIOs to the corresponding bits of an integer value. The following numbers are the movement codes for the device.

6: forwards
9: backwards
10: Turn left
5: Turn right

Additionally, 2 PWM ports controlled the duty cycle of the motors. When set to +3.3V, the motors would turn according to the logic inputs from the GPIOs, and when 0V, the motors would

not turn, Each port corresponds to the input for one of the motors. A single PWM port drove both wheels at a constant 75% duty cycle.

SystemD Startup Script

We directed the OS to run a script in a location on startup using a .service file located in etc/systemd/system as well as in lib/systemd/system. We did this by creating a symbolic link between the two locations using bash command ln. bluetooth.service, contained the information below.

```
[Unit]
After=mysql.service

[Service]
ExecStart=/usr/bin/bluetooth.sh

[Install]
WantedBy=default.target
```

We then placed the desired startup script named bluetooth.sh (as specified in service parameter) into the specified location (/usr/bin/bluetooth.sh). We also had to make the script executable using “chmod +x bluetooth.sh”. This script compiled our executable program bluetooth.c and ran it.

Next we had to enable the service we created using these commands
systemctl daemon-reload
systemctl enable bluetooth.service

To test the service without rebooting we used “systemctl start bluetooth.service” and “systemctl stop bluetooth.service”.

Beaglebone Program

The beaglebone would start up, open pins and initialize settings then wait for a user command to begin spinning and outputting sensor information over bluetooth. After scanning for a set amount of time, the board would go into a control mode, allowing the user to issue commands and control the car’s movement.

One distance sensor on the back of the car constantly read data in a for loop (with a usleep command) while the car was in spin and scan mode. This was written to the terminal at a fast

rate, while the Matlab program polled data from the terminal at a slower rate. This ensured no duplicate value readings.

User Control

Remote user control was implemented by having the beaglebone poll its terminal for an input. If the first character of the input was an ‘f’, the motors would turn forwards, ‘l’, the motors would turn left and so on. Here is a list of all the commands the board recognized.

g - go (begin spinning, beginning of operation after initialization only happens once)
f - forward
b - back
l - left
r - right
s - stop

Scanning with Matlab

Next, we wrote a Matlab script to process sensor readings. To read from and write to a serial port from Matlab, we used the “serial” command, which initializes a port with a specified location, baud rate, byte order, bit number, etc. Then, in a for loop, we repeatedly called “fscanf” to read a value from the port. We plotted the values, then smoothed the signal with a local median to reduce noise. While the original plot had a large variation in sensor readings for the same setup -- around 200 sensor units -- this variation was greatly reduced after smoothing.

Discussion

Various improvements could be made to this system. The overall goal was to use the sensors as a way to see around the car remotely. It was intended for the car to be able to be confined in a cardboard enclosure with one opening, scan a circle around itself to find the opening, and then using user input, exit through the opening.

Relying on spinning the entire car to sense surroundings is not ideal. Having a rotating sensor system on top of the board would allow a perfectly circular motion and thus better detection. The rotations would also be more predictable than the car’s whole rotation, making it possible to plot

distances in a polar graph so that the user can see exactly the proximity of objects around the car.

Conclusion

In conclusion, there were 3 main innovations required in this project: using the bluetooth module to connect to the beaglebone via UART, connecting the bluetooth module to our computer running matlab, and plotting our data in a way that would be useful to someone controlling it from another room. I believe that we produced sufficiently clear graphs to show that it could be a viable solution if the robot is guaranteed to move in a circle. We were able to get some very indicative output from sensors which we had little success with in previous labs.