Follow Me Shopping Cart

Dennis Ruto
Sofia Panagopoulou

Devpost Link: https://devpost.com/software/final-project-3opmc1

Final Project, Fall 2021
ESE519/IPD519: Real-Time and Embedded Systems
University of Pennsylvania

CX305034
DON'T TURN AROUND. IT'S THAT CART YOU ABANDONED, AGAIN.
HEY!
HEY SHARON!
CARTOONSTOCK
A CARTOON COLLECTIONS COMPANY
HA

Table of Contents

Last edited on 2020/08/02 18:18 EDT

Abstract

A follow me robot being able to get attached to shopping carts is introduced as a convenient way for everyday shopping. The robot will have the feature of following a specific human based on their Bluetooth mobile device.

Motivation

Our inspiration for this project arose from the daily inconveniences shoppers face while getting items from a shopping facility. The customers typically move around with their shopping carts across the many isles to get their produce. This is cumbersome. To solve this, we have decided to build a follow-me shopping cart to simplify the users shopping experience. With this device, the shopping cart will follow the customer making it more convenient for our user to purchase their products.

The developed device could be added to various shopping cart sizes and make real-life shopping life much more convenient. One of the challenges to be addressed is how to identify our user from the pool of all other customers within the same area. Supermarkets can get crowded plus any interaction between our user and other people should not interfere with the robot's function. To solve this problem, we will use Bluetooth and configure our robot to link with a Bluetooth enabled device e.g. cellphone, fitbit etc.

An extra feature that could get added if time permits is the addition of a remote controller determining the distance the follow-me cart is going to keep from the user.

Goals

1. Order and receive all the necessary parts
2. Build Structure of Robot
3. Develop software for the main follow-me part
4. Communication with Bluetooth and connection to specific cell phone
5. Introduction of Bluetooth function to the main follow-me code
6. Potential extra feature of remote

A. Milestone 1

Points 1,2,3 and introduction to 4th should be completed by the first milestone.

B. Final Demo

Points 4,5 and possibly 6 should be completed by the end of the semester. Also, potential corrections and improvements to the existing structure and program.

Methodology (from Milestone)

The core of the project is going to be the robot assembly in addition to the program that instructs it to follow humans. There is a lot of information online for both hardware and software above mentioned parts but the software is written in Arduino code so the understanding and conversion of the program to C/C++ is going to be a priority.
Each of the two IR sensors will be placed on each side of the robot's "head". These IR sensors can detect any obstacle (in our case humans) within a specified distance. When something is detected, the program is going to start functioning:

- The "head" of the robot containing the IR sensors and the ultrasonic will be able to turn, facing our user all the time.
- The ultrasonic detects the exact distance between the robot and the detected obstacle. This information can then be used to calculate the speeds of the four wheels.
- Using the Bluetooth module we will establish a connection between the robot and a specific cell phone. Then, depending on how strong the signal is, we will compare the calculated distance of the Bluetooth with the ultrasonic measurement. If they are matching, then the detected human is our user so the movement towards them may begin. If not, then we can ignore the ultrasonic measurements because it means that another obstacle is being detected.

Note: The reason we need both the ultrasonic and the Bluetooth is because the Bluetooth only gives us information based on a radius around the device, while the ultrasonic will be facing at the correct direction of the path.
In case we have time to complete the extra features, we are going to assemble a simple remote with two or three buttons, one IR LED (we chose IR LED so we can avoid interference caused by the overhead lights) and add another simple IR sensor to the robot. Pressing each button will make the IR LED blink with a different frequency. From the robot side, each frequency detection will translate to a different distance command to the robot. For example, for 70Hz the robot will keep a 5 feet distance from the user while for 40Hz it will keep a 2.5 feet distance.

Note: We have to keep in mind that the distance has to always be relatively small so that the user can easily reach the cart and place the items inside.
For the motion of the robot, we are most likely going to implement differential drive, which means the separation of the four wheels into two groups: the left and the right side wheels. Each pair is going to have different velocity commands which would allow the robot to make any necessary turns.

Technical Details

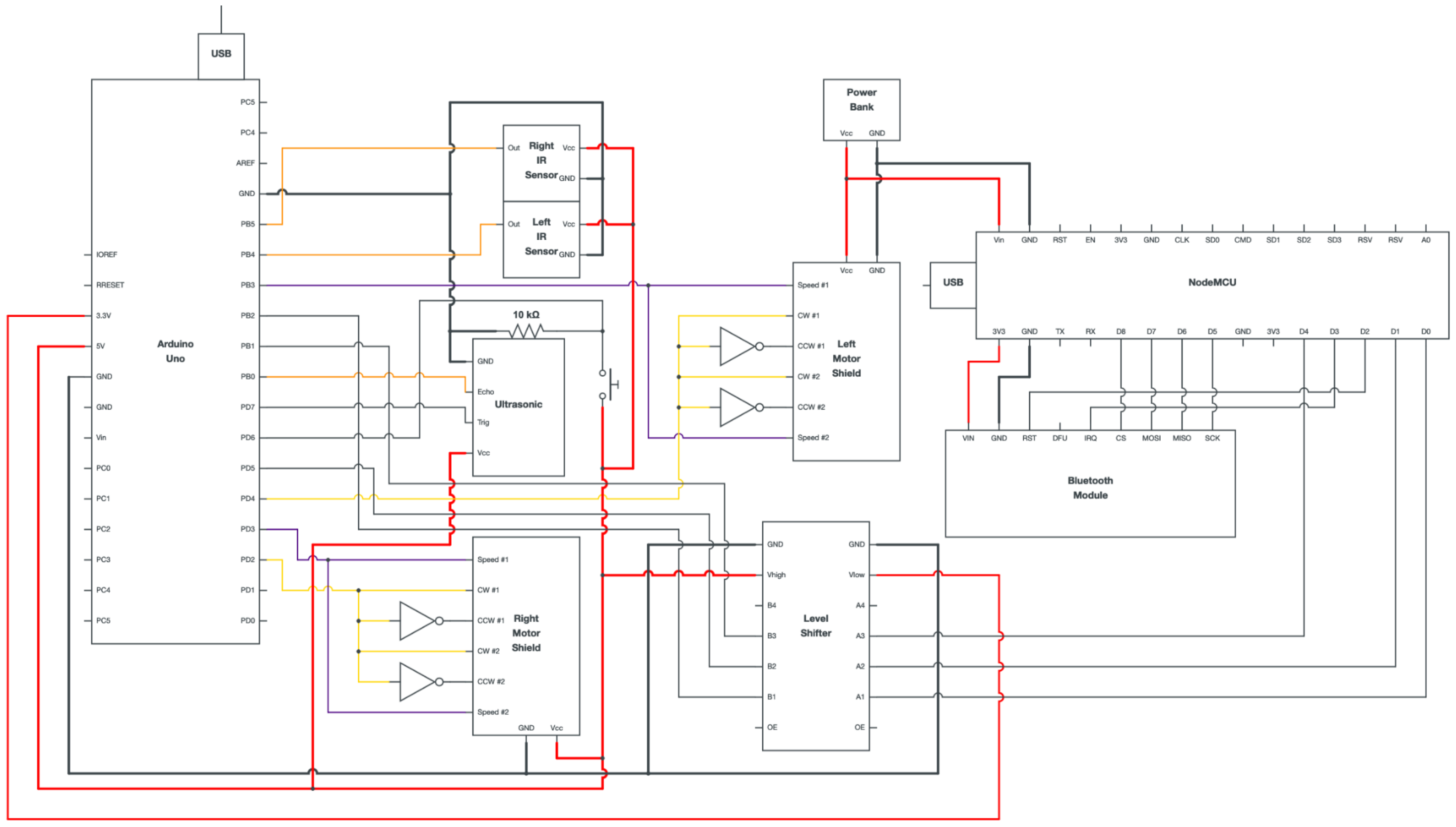
Circuitlab Link: https://www.circuitlab.com/circuit/5b6689jh2tk8/ese-519-final-project/
For documentation purposes, some important technical points of the project are mentioned below:

a) Power management:
    i) The four dc motors, when powered from the Arduino 5V, require a larger stall current than what the specific Arduino pin can provide. This results in only two or three out of the four motors moving at one time. Our solution to this problem was to use the second port of the power bank to power two motors, while the other two are powered from the Arduino 5V pin.
    ii) The NodeMCU microcontroller that was added to introduce the Bluetooth function, can be either powered from its usb port, or from its Vin pin. The Vin pin option turned out to be very useful. Otherwise, we would have to add another power bank since we had already used both ports of the first one.

b) Timers: Arduino Uno has three timers. At the beginning of the project we had added the servo motor as well, so we needed four timers in total (servo motor, right wheels, left wheels, and ultrasonic sensor). The solution to this “timer shortage” problem was to use the OCRnA and OCRnB functions of the same timer separately.

c) Pins: Arduino Uno has many available pins, but not unlimited. Each of the four dc motors requires three control pins (direction clockwise, direction counterclockwise, and speed). But using twelve pins only for the motors would be a huge waste. Our solution to that problem was to combine the left motor direction pins together (and the right ones), as well as add an inverter for the clockwise and counterclockwise direction.

d) Wheels code: The code recommendation for the left and right wheels movement mentioned one side of the wheels moving forward and the other backward, in order to make a more quick and efficient turn. After completing the robot structure, we found out that the robot was not moving at all under these circumstances. We decided to have all four wheels going forward and just provide them with different speeds. This is why the turning right and left functions are not as visible as they could.



Results

Link to Youtube video: https://youtu.be/UMGsdQIZP_E

We decided to make the following changes to the original project plan:

a) Servo: Initially, the servo would rotate until an obstacle is detected, and then depending on its angle, the code would generate the appropriate speed commands for the wheels.
We found that the use of the two IR sensors can provide us with sufficient information about the direction of the detected obstacle.

b) Bluetooth: The initial goal was to rely on the signal of the Bluetooth and calculate the distance between the module and the connected device (cell phone). This information would add another condition to the code (if something is detected and if it is the same as the connected cell phone, then follow it). Based on this information, the robot would make the decision to avoid following a random obstacle and follow the user.

However, using Bluetooth to measure distance is unreliable. According to [4], RF devices like our device rely on RSSI (Received Strength Signal Indicator) to measure the strength of the signal. The closer the distance the stronger the signal. Unfortunately, there is no useful correlation between signal strength and distance. Furthermore, interference for the surrounding as well as antenna orientation adds to the challenge of getting an accurate estimation. Our distance from the user is relatively small, 30cm. We attempted to GPS latitude and longitude from Bluetooth. This was also unreliable due to the latency of the incoming updates. Our user would have to move by more than a meter before meaningful detection could be observed.

Because of these challenges, we decided to allow the user to control the robot using their phone(just as we did in lab 4). We mapped digital pins (forward and backward) to the NodeMCU microcontroller and programmed in Arduino code to trigger whenever a change in the digital pin was detected. This control is what we aim the user to use to guide the robot as they wish, just like a manual override capability.

Additionally, the NodeMCU provided extra pins that proved useful in connecting to the Bluetooth via SPI.

The autonomous function of the robot is described in detail below.
Each of the two IR sensors is placed on each side of the ultrasonic sensor, making it feasible to determine the direction of the obstacle. Specifically, these two types of sensors coordinate as follows:

- If both the IR sensors detect an obstacle and the distance from the ultrasonic is within 8 and 30 cm, then move forward
- If only the left/right IR sensor detects an obstacle, then move left/right
- If the distance from the ultrasonic is less than 5 cm, then move backward
- If none of the above happen, stop

The individual functions for the movement of the robot are determining the speed of the left and right wheels, as well as their direction (forward or backward). Specifically, for the forward and backward movement, all four wheels have the same speed and the direction is forward and backward, respectively. For the left and right movements, all four wheels are going forward but the speed of the right wheels is larger than the left ones and smaller, respectively.

Conclusion

Technical

Overall, the main “follow me” function of the robot reached our expectations and all the work of incorporating the different hardware elements and combining the Arduino timers paid off. It was very interesting to see the individual parts of the labs being combined in one individual robot structure.

The IR sensors we used were not ideal because even after calibrating the potentiometer that is linked to the detectable distance, the obstacle detection light was blinking, indicating that the sensor is constantly detecting and - at the same time - not detecting an obstacle. It complicated the testing and the code, but within reasonable limits.

The ultrasonic sensor has a wide range of measurements, but it is difficult for it to detect something that is not a clear vertical surface. This limited our testing since we could easily use our hand as an obstacle, but not our legs when walking.

The structure of the robot is as organized as possible, with the cables being color coded, but the Arduino Uno requiring jumper wires created some visual complication (versus soldering on a perf board). Hardware debugging was much easier though, since moving the cables around took very little effort.

We believe that with better sensors, this project could easily transform into either a larger structure that would match the size of a shopping cart, or a smaller “kit” that could get attached to a shopping cart and follow the user. In the Bluetooth section, there is definitely a lot more to be done before incorporating it to the shopping cart idea.

General

We learnt that it is very important to do a lot of research and organizing before beginning the project. The power management and pin allocation are big chapters that should be taken care of before the commence of the project, to save time. Modifications can be made along the way and it is also important to be able to modify and improvise when there is an issue.

Looking back at the project process, if we got another chance, we would first do all the research about the Bluetooth function we had in mind so that we could have a better understanding of what is possible in that sector. This would save us a lot of time which at the end could be used to add a more complicated feature to the Bluetooth module - cell phone connection.

References

1. Main structure:
https://create.arduino.cc/projecthub/mohammadsohail0008/human-following-bot-070eaa
https://www.youtube.com/watch?v=yAV5aZ0unag

2. Human sensing options:
https://en.wikipedia.org/wiki/Human_sensing

3. Bluetooth and distance:
https://electronics.howstuffworks.com/bluetooth-surveillance.htm
https://smartsensordevices.com/distance-measuring-solution-for-covid-19-using-bluetooth-low-energy/
https://www.instructables.com/Arduino-Bluetooth-and-Ultrasonic-sensor-TUTORIAL/

4. AdaFruit BLE resource
Townsend, Kevin. “Introducing the Adafruit Bluefruit Le Spi Friend.” Adafruit Learning System, https://learn.adafruit.com/introducing-the-adafruit-bluefruit-spi-breakout/faq.

Appendix A

How the Adafruit BlueFruit library works.
packetParser.cpp - parse the incoming packet.
BluefruitConfig.h - maps the SPI pins
Controller.ino - main code

Node MCU SPI Pins
- CLK - D5
- COPI - D6
- CIPO - D7
- CS - D8
- IRQ - D3
- RST - D2

For our project, we utilized the controler.ino file to receive sensor readings from the Adafruit app on the user's phone. Before the Bluetooth sends data, several commands in the controller.ino file are sent to the Bluetooth.
These commands are listed below:

ble.echo(false) - Disable echo command from Bluefruit
ble.isConnected() - Ensure connection with bluetooth
ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) - To check the firmware version on the Bluetooth
ble.setMode(BLUEFRUIT_MODE_DATA) - To inform the Bluetooth device to stream sensor data

These commands are all sent in the setup function within the arduino.

The key abstraction used in the code is the Adafruit_BluefruitLE_SPI class in the Adafruit_Bluetooth_SPI.h file. This class maintains the necessary functions and pinouts needed to send and receive data from the Bluetooth. The constructor is instantiated with the SPI pins listed above(Node MCU SPI pins). The **m_rx_fifo** and **m_rx_buffer** are used to buffer incoming data. The packet size for our sensor readings is 15. The class also has a transmit buffer (**m_tx_buffer**) and a counter (**m_tx_count**) that ensures the packet to be sent is within maximum packet size limit. The class has two functions sendPacket and getPacket that send and receive packets from the Bluetooth.

The controller.ino file polls for the incoming packets in the main loop. Whenever data is received, it is parsed in the **readPacket** function. This function performs the validity check for the message using a checksum. It returns the length of the message. It is this message that we use in our project to see which digital pin has been activated. We turn the pin on/off and this is read by the arduino.