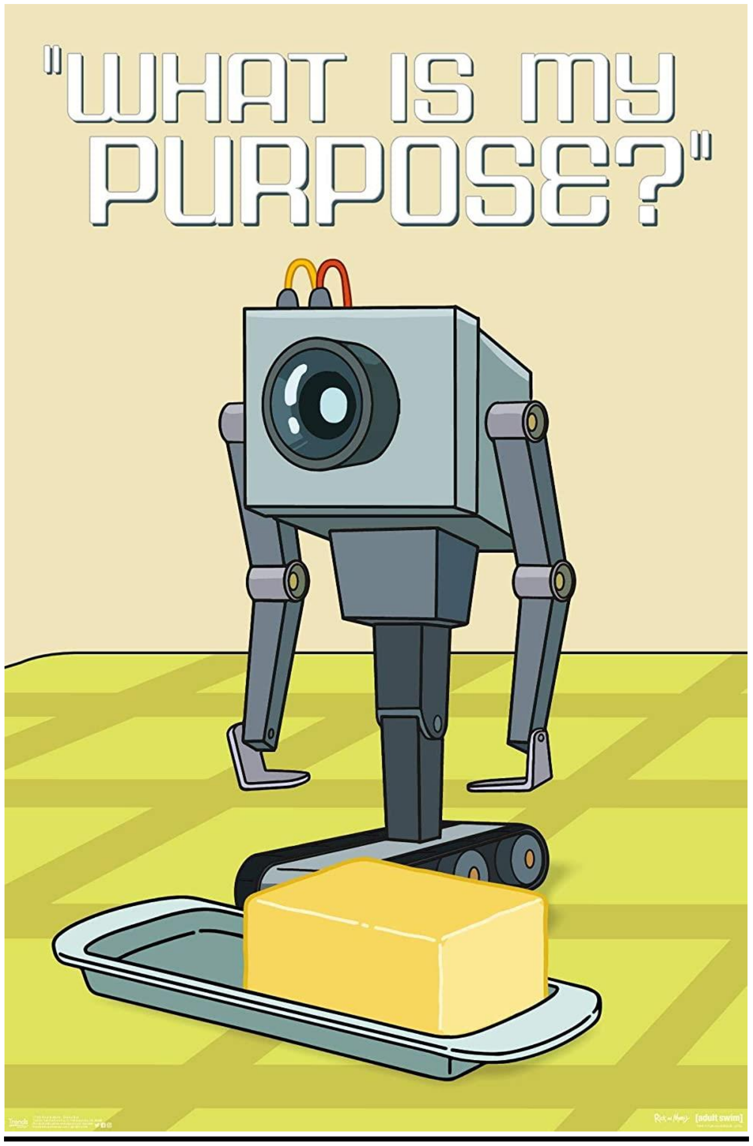
[BUTTER-BOT]

Nishant Sharma
Shashank Khemka

Devpost. GitHub. YouTube.



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

Table of Contents

Abstract

The project is an autonomous robot that searches for, and grabs the desired object, then returns it to the destination. The autonomous robot maps the area, plans a path through all objects detected, and then scans for the target object. If the scan test result is positive that means the object is detected. The robot then grabs the object using 3D printed claws. The robot then traverses to the destination and drops the object. A unique solution is implemented to solve the problem of indoor localization and navigation for autonomous systems using velostat. Once the coordinates were obtained, the bot used a path planning algorithm to traverse through all coordinates that had objects on them. The target object has a unique RFID, for which the bot scans. If the desired RFID is scanned, the robot grabs the object and place it at the destination. The robot can automate supply chains processes where goods are to be stacked/loaded like a retail warehouse.

Motivation

We were inspired by a scene from an animated cartoon series 'Rick and Morty'. The scene involves one of the main characters make a small robot that moves on dining tabletop. The robot's job is to find butter and passing it to the character. This required the robot to first identify butter kept amongst a lot of other objects lying around on the table. This idea was extended to moving/acquiring a particular known object/good placed amongst several others. Places like warehouses can make use of such automation where indoor mapping and indoor positioning for embedded integration is vital.

Goals

A. Milestone 1

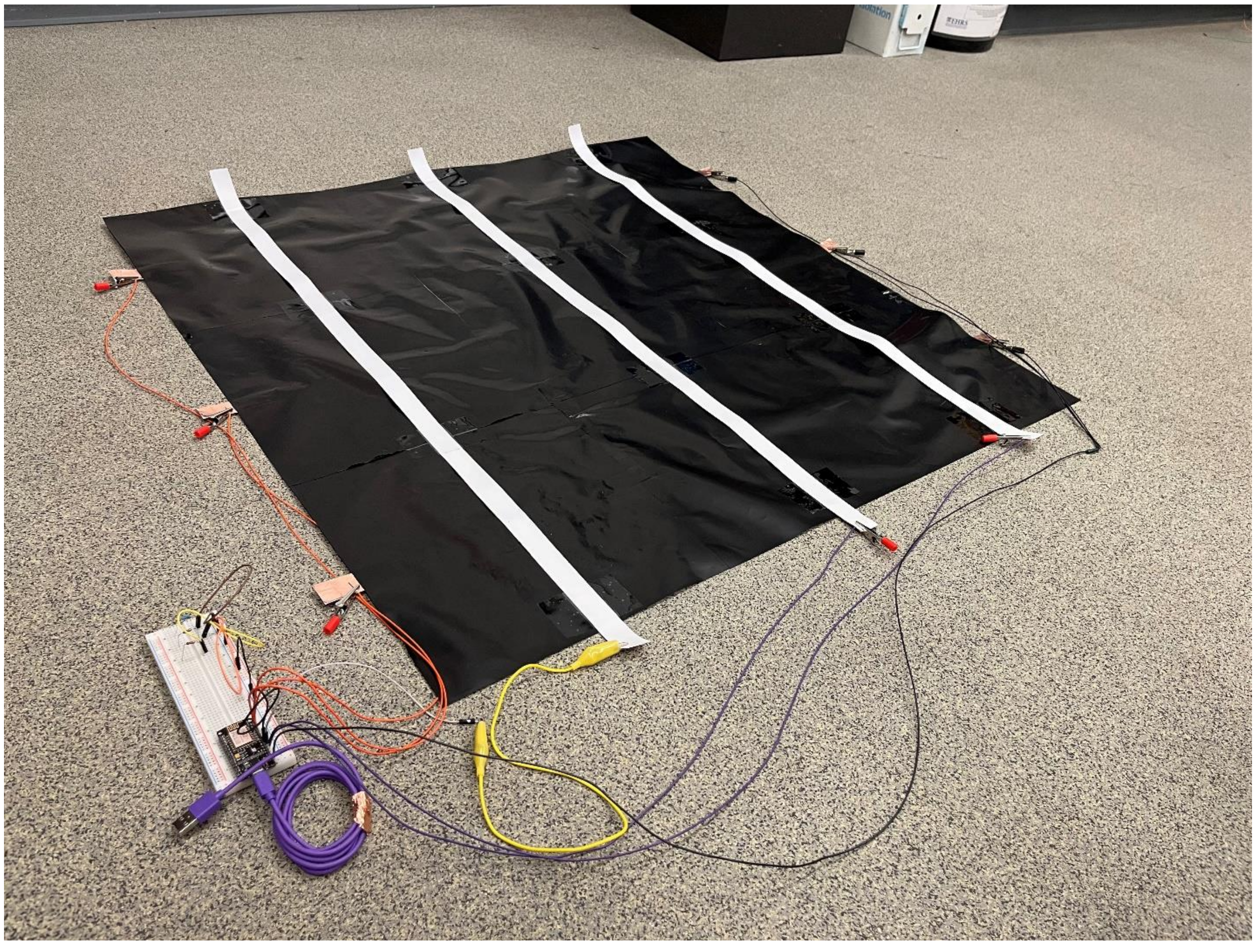
- Scanning for butter (scan for RFID tag- using RF transceiver).
- Detecting butter's direction.
- Moving towards the RFID tag.
- Grabbing hold of the butter.
- Scanning and detect the direction of the breakfast plate.
- Reaching the breakfast plate with the butter.

B. Final Demo

- Including wireless trigger remote.
- Including obstacle evasion.
- Detecting edges and avoiding falling off the edge.
- Adding audio interactions.
- 3D printing the chassis.

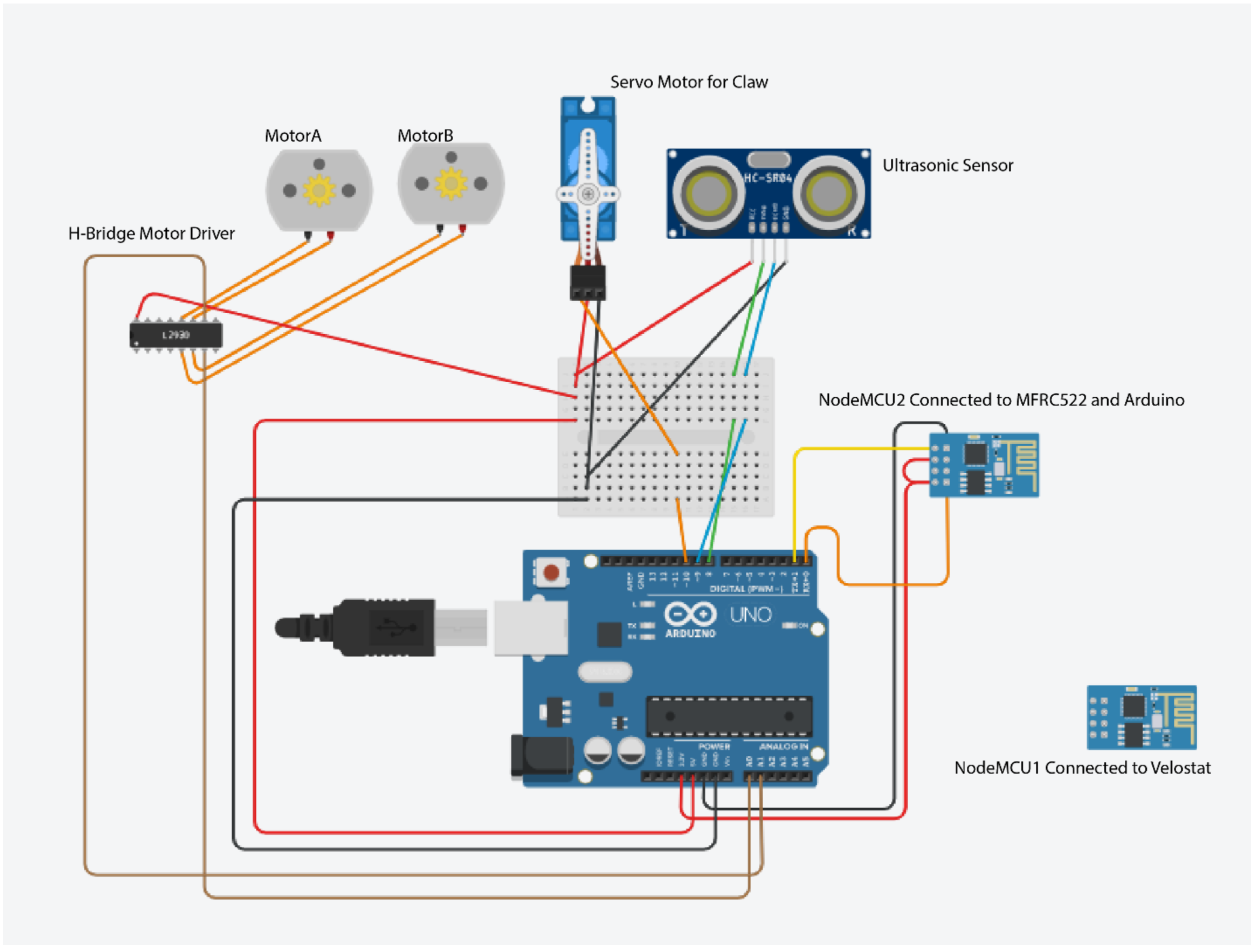
Methodology

The robot was supposed to be fully autonomous and hence it was supposed to have all the information necessary to detect the target object and drop it to a set destination. For this to achieve, pressure was sensed to detect the position of objects. Velostat was used to make a pressure sensor. Velostat is a pressure sensitive material which looks like a black paper popularly used with smart/conductive fabric products. Under no pressure the material is an insulator, but when pressure is applied on it, the resistance across it drops and the material conducts electricity through it. We exploited this property of the material to make a pressure sensitive grid. The project uses a 3 by 3 grid for demonstration purpose, but all the codes are scalable allowing for a larger grid size by just changing a few macros in the code. Copper tapes are used to make conductive grids. One side of velostat is covered with horizontal strip of copper tape and the other side of the velostat paper is laced with vertical copper tape. When pressure is inserted on the intersection of these horizontal and vertical copper tapes, the resistance across velostat drops and shorts the two copper tapes by conducting electricity through the velostat. This grid corresponds to spatial coordinates the robot is supposed to traverse. With horizontal and vertical copper tapes representing coordinate axis. Each intersection is a point. Vertical tapes are connected to output pins of NodeMCU1 and horizontal pins to input pins. Logic 1 or 3.3V are looped through each of the output pin, while one of the pin is HIGH, all the horizontal tapes are read via the input pins of NodeMCU1. If an object is placed, it would short the intersection and 3.3V would be read on the input pin of the node. This would give us the coordinate of the object.



A local network is set up between two NodeMCUs, NodeMCU1 and NodeMCU2. NodeMCU1 is reading values from the velostat. NodeMCU2 is connected to RFID tag reader and the ATMEGA 328P chip which is ultimately running the robot. NodeMCU2 acts as the server and NodeMCU1 as the client to NodeMCU2. NodeMCU2 sends data to ATMEGA 328P using UART protocol.

The robot receives grid data of the Velostat from the NodeMCU through UART communication protocol form the RX pin on the Arduino. The data comes in as a string with a series of 1's and 0's that denote if there is an object on the Grid. Then, that sequence is converted to a 2D array which stores the coordinates of all the objects. An algorithm is designed where the bot targets each object on the grid and traverses the grid to reach the target. A function traverse() is called from the search() function which is called in the main function. Inside the function traverse(), the bot updates it's x-coordinate first while also detecting if there is an object in it's way using the ultrasonic sensor with a fixed distance threshold. As soon as it reaches the desired x-coordinate or detects an object in the way, it starts updating the y-coordinate or calls a function verify() which takes in data from the RFID module which sends a logic level HIGH if the target i.e. butter is detected. After it reaches the desired coordinates, then again calls the verify() function and checks for butter. If there is a false check and butter is not detected, the bot moves towards the next target or object. Finally if butter is detected, the servo motor is triggered and the bot grabs the butter. Then again, the traverse() function is called and the coordinates of the destination are supplied to the bot. The bot reaches the destination and drops the target. The main program is a path planning conditional algorithm that updates the bot coordinates while moving it and orients it in the direction it needs to move in. There are macros created for all the directions and boundary checks so that the bot does not go out of the grid. It is a modular code that can be scaled to any grid size and for N number of objects using the Velostat grid or any other technique that supplies the coordinates of the objects placed.


Servo Motor for Claw
MotorA
MotorB
Ultrasonic Sensor
H-Bridge Motor Driver
NodeMCU2 Connected to MFRC522 and Arduino
NodeMCU1 Connected to Velostat

Component	Model
Microcontroller	Atmega328P
Sensor	MFRC522 RFID Module
Sensor	HC-SR04 Ultrasonic Sensor
Sensor	Velostat Grid (Pressure Sensitive)
Actuator	33GB-520 DC Motor
Actuator	Servo Motor SG90
Wireless Communication	NodeMCU (ESP8266)
Voltage-Level Translator	Breakout - TXB0104

POWER MANAGEMENT

9V batter- to power arduino through the barrel jack.
9V batter- to operate DC motors rated upto 12V.
5000mAh power bank- to power NodeMCU2.

Results

A fully autonomous robot was created that would scan for and drop the desired target object to a set destination. The robot is capable of autonomously traverse through objects/obstacles. The final design included a mat made from velostat material which was used to map objects onto a coordinate grid. This information was sent to the robot wirelessly via a access point connection. This includes communication between two nodeMCUs and then serial communication between one of the nodes and the Atmega328P, which runs the robot. After receiving the positions of all the objects placed on the grid, Atmega328P plans a path to move across the coordinate plane. Once the target is detected the robot claws it and drops it at the destination. If there are multiple objects on the grid, and the robot finds the target, it would not further traverse to scan other objects. The robot then simply moves towards the set destination.

Conclusion

The project involves execution of a lot of concepts. There are multiple systems coming together to become this one autonomous robot. There was a lot of research and brainstorming put in to come up with localization and indoor navigation. While researching about the methods to localize the target object we learned a lot about different kinds of sensors and their limitation. Codes were written such that they are all scalable and modular. Velostat mat was a unique solution to localization which otherwise is solved using more complex sensors and processors. This concept can easily be scaled up to SLAM(simultaneous localization and mapping). Some of the challenges faced that were unanticipated were that the copper tapes that we used did not conduct on the adhesive side even though it said it uses conductive adhesive. 16 timer was required twice. Working with only 1 16-bit timer was challenging. The tank chassis tread had a physical manufacturing defect which resulted in it being not balanced. This results in the right tread being faster. The range of the RFID reader is very less and this required us to plan the position and design of the claw and RFID reader. Next step for this project could be to add voice command to start searching for objects. Instead on searching for 1 object with RFID, it could have multiple objects with unique RFID tag and depending on what it wants, it could search for the corresponding RFID. For example, there are different colored boxes with unique RFID corresponding to different colors and the bot could recognize and pass different colored boxes based on the RFID. This would represent different packages in a warehouse all having different RFID.

TOPICS COVERED

Timers: Timer1 is a 16 bit timer. it was used to generate PWM trigger signal of 20ms for servo motor. 1ms PWM for 0 degree rotation of the servo motor and 2ms for 90 degrees. Timer1 is also used for ultrasonic sensor to calculate the time taken for the signal to echo back. This time is used to calculate the distance between the robot and an approaching object.

Interrupts: Pin change interrupt is used to detect echo of the ultrasonic sensor and ISR is used to copy data coming in from NodeMCU2 via UART.

Serial communication: SPI and UART serial communication protocols were used in the project. The MFRC522 RFID reader communicates using the SPI protocol. NodeMCU2 uses UART to send the coordinate data to the Arduino.

Wireless communication: wireless connection was set up between two NodeMCU via the ESP8266 chip. ESP8266 uses TCP/IP protocol for communication. Access point is set up and the host acts like the server and the node sending the data as the client.

Path planning: The path planning algorithm for the robot works on the principle of relative distance of the robot from the target object. The robot calculates the distance of its current coordinates from the target coordinates. A function takes in the data and makes the robot move in the direction of the target by changing its orientation to align it towards the target and updating its coordinates. The alignment of the robot is taken care of by a function that takes in the current orientation and compares it with the target orientation according to the macros and turns the robot in the right direction continuously until the target orientation is reached.

3D printing: 3D printed the claw.

References

https://www.arduino.cc/en/Reference/WiFiStatus
https://www.arduino.cc/en/Reference/ClientConnect
https://www.arduino.cc/en/Reference/ClientPrint
https://tttapa.github.io/ESP8266/Chap07%20-%20Wi-Fi%20Connections.html
https://www.hackster.io/news/yoga-mat-size-pressure-sensor-matrix-uses-lattepanda-and-velostat-to-save-money-49ee86950e5a

Appendix A

ESP8266WiFi.h:

WiFi status() is function is used to check the status and then proceed to the further program.

WiFi.status()- Return the connection status.

Returns
WL_CONNECTED: assigned when connected to a WiFi network;
WL_NO_SHIELD: assigned when no WiFi shield is present;

WL_IDLE_STATUS: it is a temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (resulting in WL_CONNECT_FAILED) or a connection is established (resulting in WL_CONNECTED);
WL_NO_SSID_AVAIL: assigned when no SSID are available;
WL_SCAN_COMPLETED: assigned when the scan networks is completed;
WL_CONNECT_FAILED: assigned when the connection fails for all the attempts;
WL_CONNECTION_LOST: assigned when the connection is lost;
WL_DISCONNECTED: assigned when disconnected from a network;

connect() Connects to a specified IP address and port. The return value indicates success or failure. Also supports DNS lookups when using a domain name. In our code it connects to port 80.

client.print(data) Print data to the server that a client is connected to. Prints numbers as a sequence of digits, each an ASCII character (e.g. the number 123 is sent as the three characters '1', '2', '3'). Prints the values from the velostat to the access point.

MFRC522.h:

MIFARE is the NXP Semiconductors-owned trademark of a series of chips widely used in contactless smart cards and proximity cards. According to the producers, billions of smart card chips and many millions of reader modules have been sold.

PCD_Init()- initialize all the register to default values.
Set the chipSelectPin as digital output, do not select the slave yet
If a valid pin number has been set, pull device out of power down / reset state.
First set the resetPowerDownPin as digital input, to check the MFRC522 power down mode.

PCD_DumpVersionToSerial()- Dumps debug info about the connected PCD to Serial. Shows all known firmware versions. It receives no arguments and also returns void. This method will print the information to the serial port for us, which is why it returns void.

PICC_HaltA()- Instructs a PICC in state ACTIVE(*) to go to state HALT.
Returns
STATUS_OK on success, STATUS_??? otherwise.

rfid.PCD_StopCrypto1()- Used to exit the PCD from its authenticated state. Remember to call this function after communicating with an authenticated PICC - otherwise no new communications can start.

PICC_IsNewCardPresent()- Returns true if a PICC responds to PICC_CMD_REQA. Only "new" cards in state IDLE are invited. Sleeping cards in state HALT are ignored.

Returns-
bool

PICC_ReadCardSerial()- Simple wrapper around PICC_Select. Returns true if a UID could be read. Remember to
call PICC_IsNewCardPresent(), PICC_RequestA() or PICC_WakeupA() first. The read UID is available in the class variable uid.

Returns-
bool