BreakfastBot

Patrick Wang
Katarina Bulovic

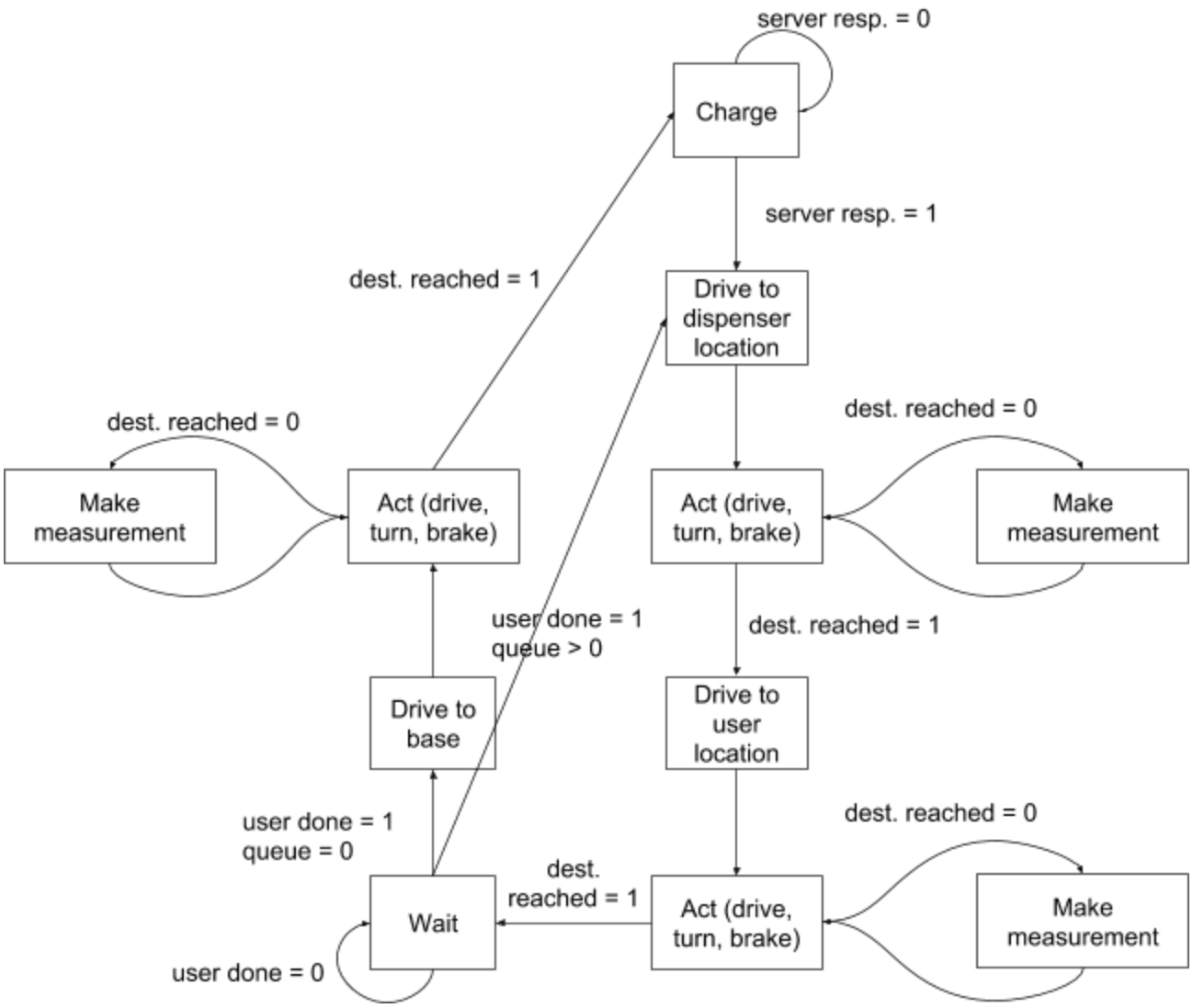
Overview:

We want to build an autonomous robot that remotely delivers snacks to a user. The robot will be called to a user's location based on input to a website or app. It will travel to a snack dispenser, collect food based on what the user requested, move to the user's location to deliver the snack, then returns to its base. The website will have inputs for desired type of snack and current user location, and should be accessible from a computer or smartphone.

Intended Functionality:

The breadboard on the robot will include a motor driver, three distance sensors, an IMU, a 9V battery to power the motors, and a power management board with a lithium ion battery to power the ESP32. The robot starts charging at its base, constantly making GET requests to the server. When the server responds that a user has requested snacks, the robot travels to the snack dispenser and physically pushes a lever/button that dispenses snacks (or potentially, we can attach ESPs to the dispensers, and control dispensing via WiFi). The robot then drives along the hallway, making distance and accelerometer measurements to control itself to the correct location. Upon arriving, it waits for the user to pick up the food and receives some input from the user (Voice? Button press?) that tells it to return to base. The robot then drives back to its base, controlling itself again with sensor measurements.

The UI should include information about availability of snacks and delivery time. Quantity of snacks left as well as statistics can be stored in a database.



Predicted Technical Challenges:

- The robot will need to make constant GET requests in order to respond to user queries quickly. However, this will use up battery very quickly. Possibly use Wake on LAN?
- The robot does not have perfectly accurate location nor does it have perfect knowledge of where it is on the map. However, it will need to drive precisely beneath snack dispensers and onto the charging dock.
- We will somehow need to charge both the 9V alkaline and 3.7V Li-ion battery.
- The robot must recover from being dislocated or knocked over.

Parts List:

Item	Price	Qty	Vendor	Item number	Description
Chassis	$22	1	Amazon	B06VTP8XBQ	Chassis kit including four motors, four wheels, and frame
Motor driver	$6.60	2	Amazon	TB6612FNG	Uses logic from microcontroller, outputs motor-driving voltage
Distance sensor x5	$9.79	1	Amazon	HC-SR04	Ultrasonic sensor with 0.3cm resolution
Breadboard	-	1	owned	-	
IMU	-	1	owned	MPU9255	IMU with accelerometer and gyroscope (from lab)
Power mgmt board	-	1	owned	ADP5350	Charges Li-ion battery with ESP or powers ESP with battery
9V alk. battery	-	1	owned	-	
3.7 Li-ion battery	-	1	owned	-	

Milestone/Demo Table:

Date	Milestone	Demo
April 19	Put together all circuit elements (esp32, motors, motor driver, distance sensors, IMU, and power management board)	Show completed circuit along with circuit diagram
	Create a “Robot” class to be used for driving the robot forwards and backwards and turning using the motors	Demonstrate example program where the robot can drive forwards, backwards, and turn
April 24/26	Make a simple website user interface that can send inputs to the server for user location and desired snack (and a temporary input field with simple commands for testing new features as we build the robot)	Demonstrate user interface with working inputs

	Write server-side code to respond to user input on the website and interface with the esp32	Show robot responding to input on the website
May 1/3	Write code for the distance sensors and IMU so that the robot can keep track of its orientation during turns and see walls and doorways in front and on either side of it - count doorways until it has gone the desired distance, turn when it gets to the end of a hallway	Demonstrate working program if there is an area like a hallway with open doors on the sides, otherwise show a video of the robot driving until it counts the right number of doorways and then stopping
	Optimize the robot's power usage by turning off circuit elements when they are not in use and possibly figuring out how to recharge the battery when the robot is at its base station	Explain steps taken and demonstrate code changes which optimize the robot's power usage
May 8/10	Build snack dispenser using boxes to be filled with snacks, motors to open/close boxes, and either a lever/button for the robot to press or an ESP32 to control the motors (if we use the ESP32, snacks could be dispensed based on server input)	Demonstrate working snack machine with dispensing functionality either when button is pressed or when requested via WiFi
May 15/17	Integrate different parts of the code so that the robot (following website user input) presses buttons on the snack dispenser to collect a snack, then gets to the right place using code from May 1/3, then returns to base station (again with May 1/3 code)	Show a video of the robot working based on user input