Embedded Systems Final Project Design Document
Fuschia Rover

Authors: Jordy A. Larrea Rodriguez, Cody Argyle, and Nicole Sundberg

Base Requirements: UART (communication with board), ADC (IR Sensor reading conversion), PID (motor control), Interrupts, Timers, GPIO

Components: 3 IR (GP2Y0A21YK0FIR) distance sensors, 2 motors, 1 20mm bearing wheel, 2 65mm diameter wheel, 1 breadboard, 2 custom H-bridge motor controllers

Bill of Materials

Component	**Quantity**	**Price**	**Documentation/link**
Purple Aluminum Chassis for TT Motors - 2WD	2	$14.95	https://www.adafruit.com/product/3796
20mm Height metal caster bearing wheel	2	$3.90 ($1.95 per)	https://www.adafruit.com/product/3948
Thin white wheel for TT DC gearbox Motors - 65 mm Diameter	4	$6.00 ($1.50 per)	https://www.adafruit.com/product/3763
Ultrasonic Ranging Module HC - SR04	3	$0 (already have)	https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
breadboard	1	$0 (already have)	NA
Lipo battery	1	$0 (already have)	NA
Custom H-bridge motor controller	2	$0 (already have)	NA
IR distance sensor (10-80cm, 4-32”)	3	$45	https://www.mouser.com/datasheet/2/737/gp2y0a21yk_e-1915144.pdf
(SHIPPING)		~$14.00	
Total Cost per person		$24.34	

Power, Sensing, Control, Motion

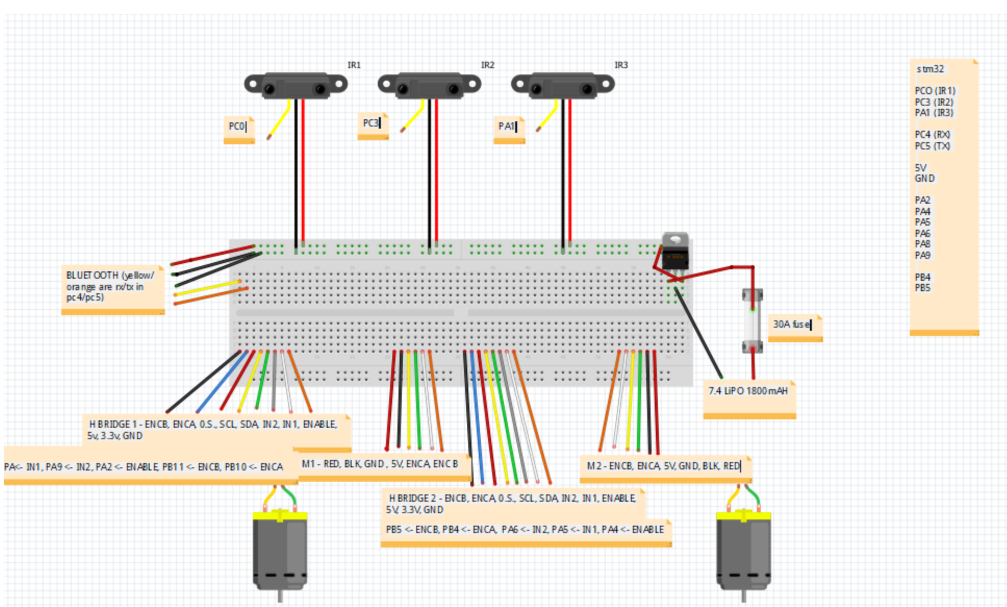
Power: Procured a 7.4 V 1800 mAh (~13.4Wh) LiPO battery to power motors, sensors, and board. Added a 20A fuse between the positive terminal of the battery and power lines of the breadboard to protect components against unexpected surge currents or short circuits. Created a 5 V low power supply via an LM7805 5 V fixed voltage regulator with a maximum current output of 1.5 A, according to specifications. A 10 μF capacitor was placed at the power lines parallel to the voltage supplies for decoupling purposes. Our motors consumed ~24Wh (~6 V at ~2 A) while stalled; hence, our power design (sensors and discovery board consumed negligible power relative to motors).

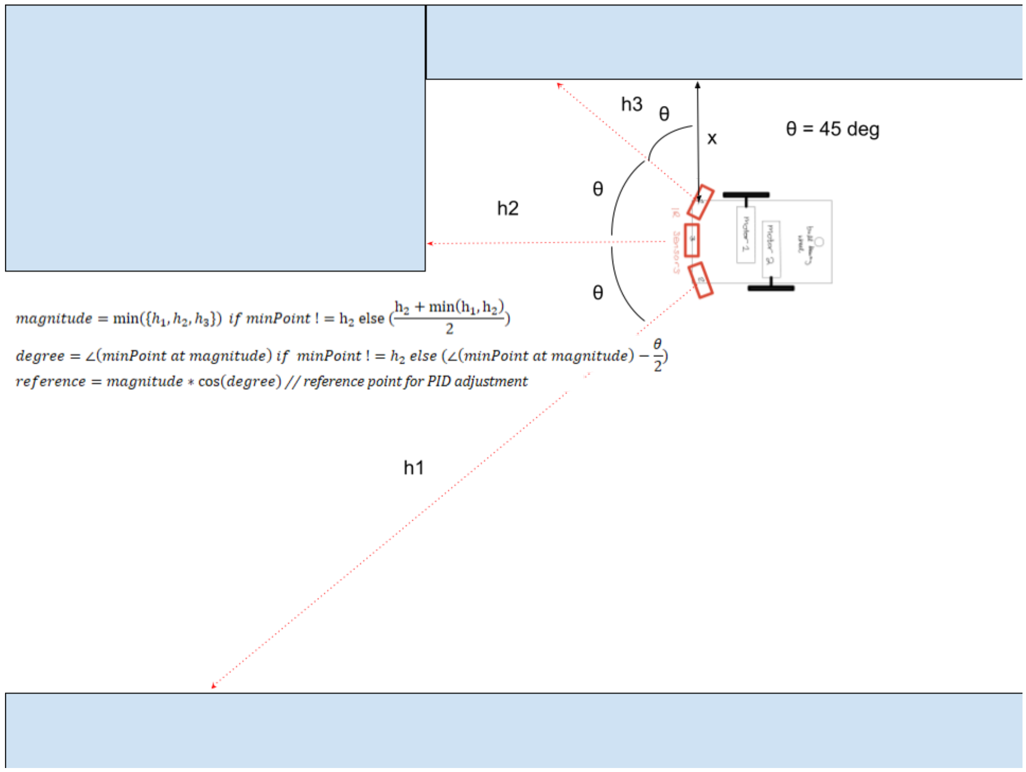
Sensing: The rover uses IR distance sensors in order to determine how close it is to obstacles or walls. These sensors will be placed towards the front of the rover, with one sensor sensing directly in front of the rover and the other two placed at 45 degree angles to sense diagonally from the robot. Data from IR sensors will inform our wall following PID.

Control: Our rover will work autonomously. We worked on manual controls via a UART connection but ultimately decided not to include it in our final design. For autonomous control, our robot will use the sensor data to determine if it should continue forward or follow a wall. The 2WD vehicle leveraged PID to maintain ~18cm from some wall or object as detected by our sensors. When all points are a certain distance from the rover, the state transitions to a state that changes the PID function to one that adjusts the motor individually in reference to the particular motor's encoder counts.

Motion: Our rover uses two motors attached to 65mm diameter wheels. We also utilize a trailing bearing wheel in the back for increased stability. Our two motors and wheels will be responsible for driving motion, allowing for forwards, backwards, and turning capabilities.

System Design:

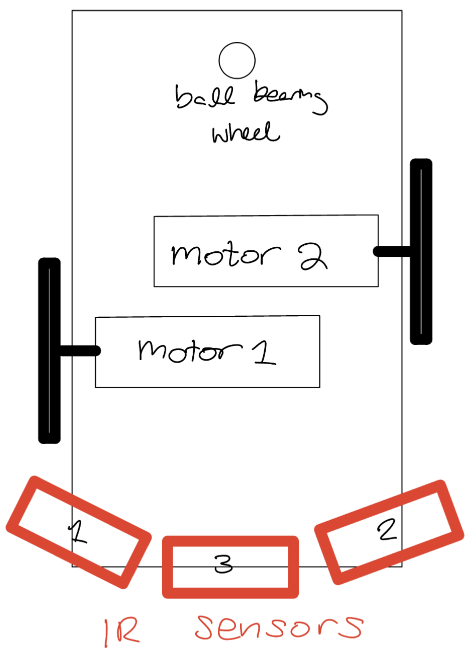




$$magnitude = \min(\{h_1, h_2, h_3\})\ if\ minPoint\ != \mathrm{h}_2 \text{ else } \left(\frac{\mathrm{h}_2 + \min(\mathrm{h}_1, \mathrm{h}_2)}{2}\right)$$

$$degree = \angle(minPoint\ at\ magnitude)\ if\ \ minPoint\ != h_2\ else\ \left(\angle(minPoint\ at\ magnitude) - \frac{\theta}{2}\right)$$

$$reference = magnitude * \cos(degree)\ //\ reference\ point\ for\ PID\ adjustment$$



Def GeneralPID:

$error = target - reference$

$proportional = k_p * error$

$integral \mathrel{+}= k_i * error$

$$derivative = k_d * \left(\frac{\Delta error}{\Delta t}\right)$$

$output = proportional + integral + derivative$

....

If state ***is*** *FollowWall:*

output = GeneralPID(target = 18 cm from wall, reference = $magnitude * \cos(degree)$*)*

LeftMotor.setSpeed(LeftMotor.targetSpeed $\pm$ *output)*

RightMotor.setSpeed(RightMotor.targetSpeed $\mp$ *output)*

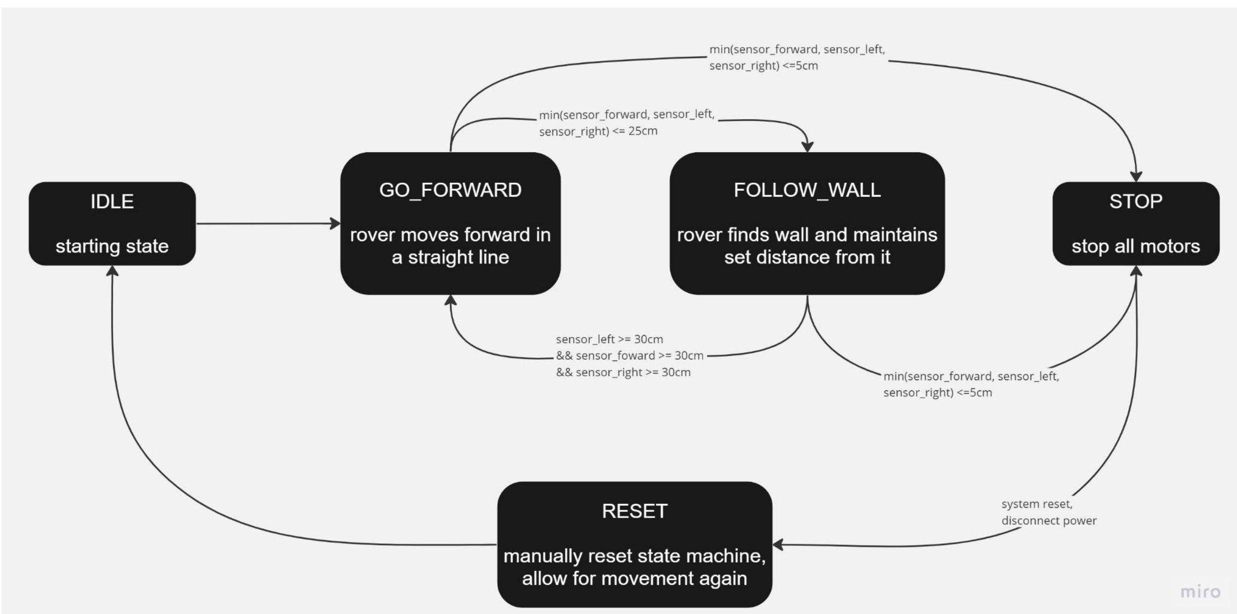
Else if state ***is*** *GoStraight:*

output = GeneralPID(target = LeftMotor.targetSpeed, reference = $RPM\ from\ encoder$*)*

LeftMotor.setSpeed(output)

output = GeneralPID(target = RightMotor.targetSpeed, reference = $RPM\ from\ encoder$*)*

RightMotor.setSpeed(output)



Milestones / Plans:

Milestone	Date	Details
Milestone 1	3/17/23	Have full design finalized, order relevant parts
Milestone 2	3/24/23	All components must be on hand and ideally assembled.
Milestone 3	3/31/23	Mount wheels and motors, have motors control wheels - forwards, backwards, turning. GPIO implemented. Power system finalized

Milestone 4	4/7/23	Read data from ultrasonic sensors, mount to our frame Have UART connection working, be able to send and receive data
Milestone 5	4/14/23	Link data read from IR sensors to motor controls, maintain a distance away from a wall. PID implemented.
Milestone 6	4/21/23	Extended Testing and Fine Tuning

Risks, Unknowns, Problems Faced:

We did face some issues with using the ultrasonic sensors and ended up changing our sensors to IR distance sensors. We utilized a bluetooth UART connection which required a connection using a linux machine and required some workarounds. The bluetooth connection setup and commands will be included in the readme, but are not part of our final implementation. We were able to use all of the IR sensors without too much interference between our sensors, so our concerns about interference weren't realized.

A major problem we ran into was the shaft diameter for the motors was significantly different than the wheel center bore. We tried using a locking screw to hold the wheel in place, but this resulted in extremely wobbly rotations under very little power, and the wheel shearing off under higher power. This led us to ordering separate wheels, only one of which arrived on time. This led us to ordering generic adapters, which were too generic. The end result is extremely fragile and consists of zip ties holding the wheels to the adapter which is held to the motor shaft by two very small locking screws. The locking screws are too small to hold at any significant motor power, and too easy to strip when attempting to tighten them more. Looking back, we should have bought smaller motors from the adafruit kit, even though they didn't have built-in encoders. Had we bought the full kit and not used the lab motors, I think we could have made this work quite well with far fewer wires.

References:

- https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- https://www.adafruit.com/product/3796

- https://www.adafruit.com/product/3777#technical-details
- https://www.adafruit.com/product/3948
- https://www.adafruit.com/product/3763
- https://www.amazon.com/SongHe-KY-040-Encoder-Development-Arduino/dp/B087ZQLLWQ/ref=sr_1_5?crid=31KQTFPW75XB6&keywords=rotary+encoder&qid=1677728661&sprefix=rotary+encode%2Caps%2C133&sr=8-5
- https://www.adafruit.com/product/164#technical-details
- https://www.mouser.com/datasheet/2/737/gp2y0a21yk_e-1915144.pdf
- https://docs.ros.org/en/humble/index.html
- https://navigation.ros.org/tutorials/docs/navigation2_with_slam.html
- https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html