Mini Rover Final Report

Alfonso Monroy - RID: 821534240 - Email: amonroy3206@sdsu.edu

Ismael Chavez - RID: 819113653 - Email: ichavez1191@sdsu.edu

Charles Kaui - RID: 818404386 - Email: ckaui3713@sdsu.edu

Description: Our device is a small autonomous rover that would go on by itself without any user input and will avoid any objects that are in front of it. This device is a cyber-physical system because it is a project that depends on both physical and software aspects to make the device work. The physical aspects of this device are the motors and the sensors, where the sensors give an input to our device to determine what movements it is to make, and the outputs are sent out to the wheels so it can move according to the signals it receives from the sensors. The software aspects are the pid controller, and the FSM that we wrote code in order to control our device.

Modeling:

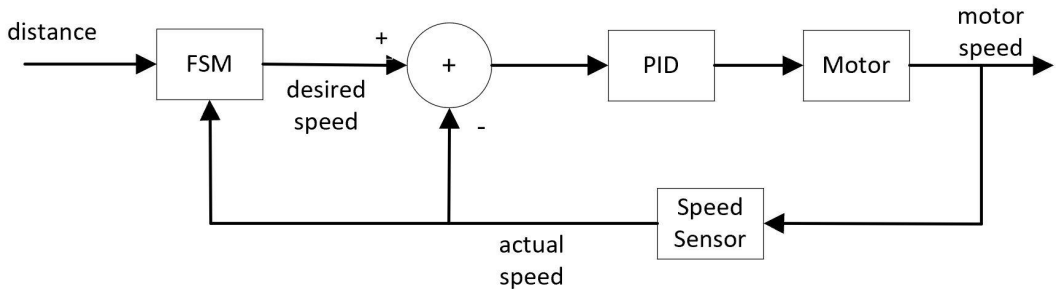

Figure1: Feedback Control Loop

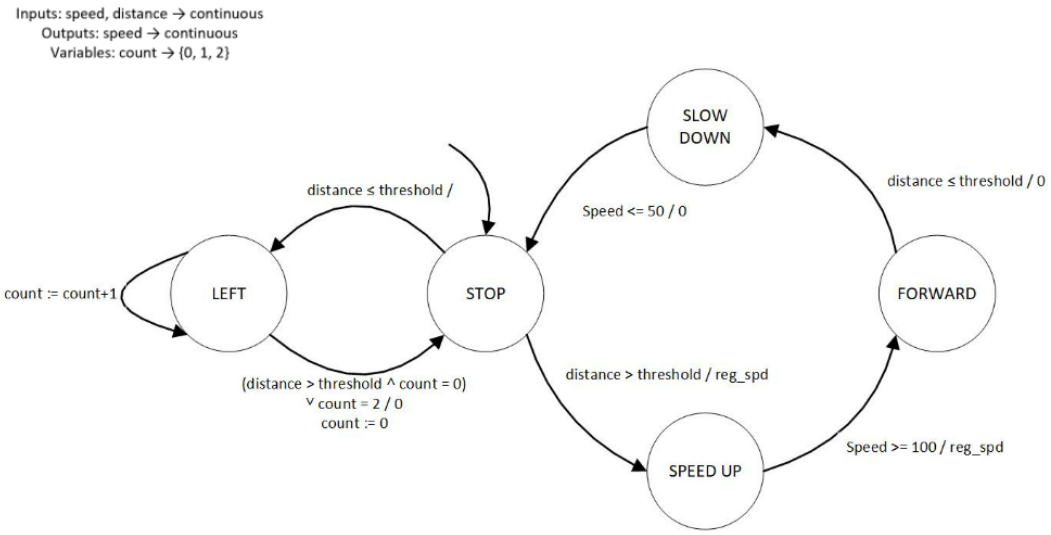

Figure 2: Finite State Machine

FSM Notation

States = {STOP, FORWARD, SLOW DOWN, SPEED UP, LEFT}

Inputs = {speed, distance -> $\mathbb{R}$}

Outputs = {speed -> $\mathbb{R}$}

initialState = STOP

Update = STOP	if (s = SLOW DOWN ^ speed ≤ 50) v (s = LEFT ^ (distance > 15cm ^ count = 0) v count = 2)
FORWARD	if (s = SPEED UP ^ speed >= 100)
SPEED UP	if (s = STOP ^ distance > 15cm)
SLOW DOWN	if (s = FORWARD ^ distance ≤ 15cm)
LEFT	if (s = STOP ^ distance ≤ 15cm)

Note: The FSM's input speed is the PWM value used to change the speed of the motors, with the maximum value being 255.

Design:

The frame of the rover was designed using Fusion 360 and 3D printed. The model components of the frame were designed by Charles Kaui using amazon rover kits as inspiration. 3d models of the electrical components were downloaded from Grabcad Community to assist in the PCB design process. The PCB was also designed by Charles Kaui using fusion 360 and printed in the senior design lab.

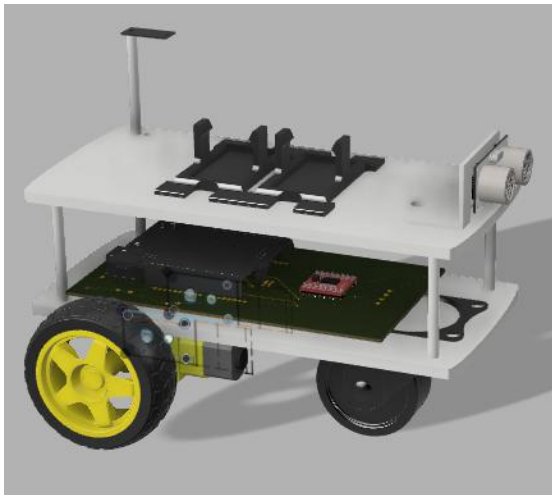

Figure 3: CAD Design

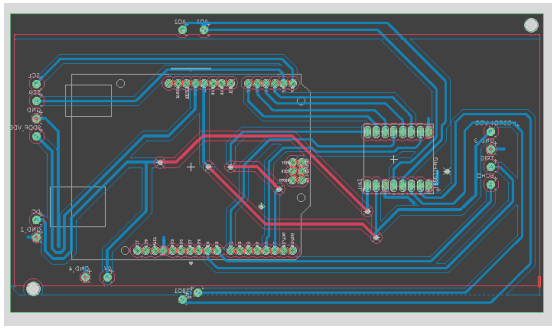

Figure 4: PCB Design

HW changes throughout the project: Throughout the process of building and making the project we were initially going to use a atmega2560 as our microcontroller, but after seeing that the microcontroller had much more pins than necessary and that it was taking a lot of space in our design we decided to go with a smaller microcontroller that we had on hand in the arduino

uno. We also changed the batteries to power our system from 4 AA batteries to 2 9v batteries because the 4 AA batteries were not delivering enough power for the system to run the motors. We also added a pcb board to the design because it was less messy and took less space then running it with a breadboard. The CAD design also changed, it initially had only 1 layer to place all of the electrical components, and then we added a second layer so we had more space for all of the components. We also acquired replacements for the motors because some of them were faulty after a few tests.

List of HW Components:

Part Type	Part	Description/Reason for choice of part
Sensor	HC-SR04	Ultrasonic sensor for distance measurement
Sensor	LSM9DS1	9 degrees of freedom which includes magnetometer to detect direction and accelerometer to detect the acceleration of the system
Microcontroller	Arduino Uno 328p	A small microcontroller since we did not need too many pins and wanted access to analog pins in case we needed them for our sensors, which we used for the LSM9DSI sensor.
Motor Driver	TB6612FNG	We picked this motor driver due to its ability to drive two motors at the same time.
Motor	2x 1568-1645-ND	Inexpensive and simple DC motors to move the device
Batteries	2x 9V batteries	Used 2 9v batteries in parallel to power the whole system.
PCB	—	Unique design for the project created by Charles Kaui and with the help of Mark Bruno through fusion 360.
CAD	—	CAD designed by Charles Kaui through fusion 360.
Wheels	2x Sparkfun wheels	To be driven by the motors for the movement of the system.
Wheels	Castor Wheel	To maintain balance of the system and to use less motors to make the same movements.

List of SW Components:

note: All of the software components are written using C, and with the arduino IDE. All implementations of the code, and integration with the hardware, was done with arduino libraries. Some of the code was not able to be fully tested due to complications with the motor driver.

Software Component	Description	Implementation
PID	Code to adjust the speed of the motors in real time.	The implementation was expected to use the Ziegler-Nichols method to compute Kp, Kd, and Ki. Acceleration measurements were implemented using the SparkFunLSM9DS1 library.
Motor Control	The code used to control the motors.	This was implemented using the arduino's SparkFun_TB6612 library for the specific motors we used.
FSM	Code to control the rover's behavior in response to distance and speed measurements.	Implemented in C using a *switch* statement and integer variables for each state. Distance measurement was done through a function modified from the SR04 library.

Analysis:

LTL Equations:

p = distance ≤ 15 cm
q = distance > 15 cm

1. **G**((s=FORWARD ^ p) => **F**(s=LEFT))
2. **G**((s=LEFT ^ q) => **F**(s=FORWARD))

The trace that represents the first equation is shown below in Figure 5. For the second equation, there are two possible traces, which are shown below in Figures 6 and 7.

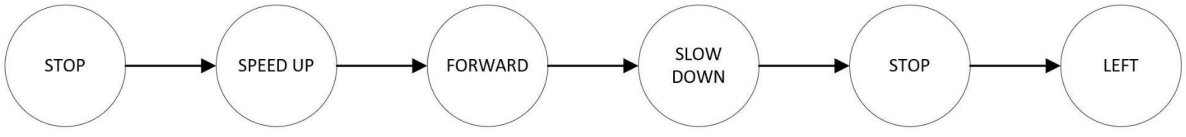

Figure 5: Trace for Temporal Equation 1

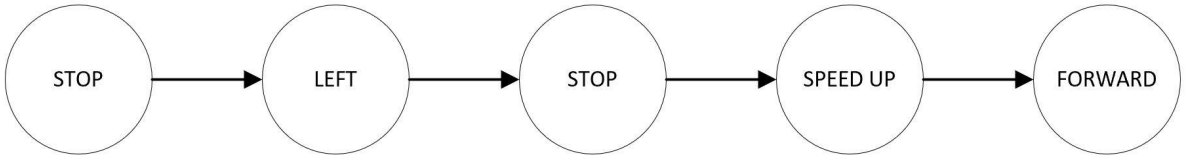

Figure 6: Trace for Temporal Equation 2

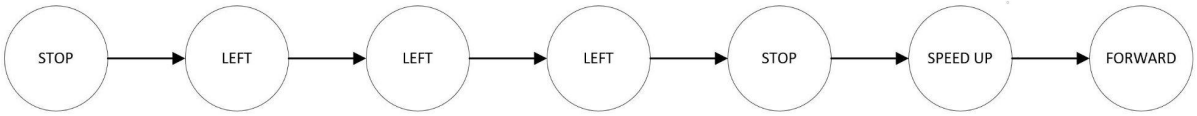

Figure 6: Trace for Temporal Equation 2

Quantitative: After running the code multiple times and calculating how long it takes for the device to run 1500 state changes on 10 separate occasions we calculated that each loop takes approximately 2.4 ms to run. The WCET in this process was 2.72 ms, which is acceptable since the change of states is not racing against any other processes, since there is only ever one process to run, it is also enough time for it to detect objects and start changing states so it can avoid them.

Metrics of Success: Our first metric of success was that our device was able to run by itself without any assistance. We were able to meet this goal fairly well since when we ran our first couple of tests it was able to move by itself until we powered it off by removing the batteries. The next metric of success was to run autonomously and not collide with any walls. We partially met this goal, but due to hardware failures of the system we were not able to ensure that we met this goal. Our next metric of success is to control the speed of the device using a feedback loop using the measured speed and the desired speed to calculate the error and respond accordingly. This metric was not achieved because of failures with the motor driver that did not allow us to control the speed of the motors. The final metric of success is a stretch goal of mapping out an area, but we did not have enough time to implement and we will talk about in depth in our extension section.

Results: The results of the project were both a success and a failure. The device was a CPS system that was capable of running autonomously and evading objects. Problems with our motor driver ended up with us not being able to control the system using the pid we wanted to implement, and there were still some collisions when it was running, while it was able to evade most objects, this was also not fully successful and would sometimes still collide with objects and there were situations where it would still collide with objects, specifically if it was not able to detect more wall like objects.

Extensions: For changes we would like to apply at a later day one of the changes we want to add multiple sensors, around 3-5 more ultrasonic sensors for two reasons. One of the reasons we want to add some of those sensors is to increase redundancy in the system. The second reason is to be able to sense if there are objects to the side for it to be able to incorporate some additional states so that it can make more efficient turns as it will always turn in an unimpeded direction. We would also want to take more time to fix the issues we had with our motor drivers, whether it was a power issue where it was not getting enough voltage or current and thus not giving the correct amount of power to the motors, or if the motor driver was faulty we were not sure towards the end. The last design change is that we would like to add a switch to turn the device on or off.
The other extension is more for giving the device a purpose, and that purpose is to add a mapping feature to the device. We will do this by creating a class called square, and make a 2 dimensional array of this class where each side is true or false, essentially start it out with the edges of the array as a true value, and have the device go around the whole area until it maps the whole area into the two dimensional array.

Missing Milestones: One major milestone that we missed was that the PID controller was not able to be incorporated with our system due to a faulty motor driver, preventing the slow down of one of the motors. One milestone that was partially missed was running autonomously and evading objects. It was partially met where it will successfully evade objects, but there were scenarios where it will just turn and bump into an object and just stay stuck there. This is one of the major reasons why we would want to add sensors to the sides so that it can make better turns and not get stuck.