Summary of Week Two

We did not get quite as far as we would like for week two, but we are still doing fairly well.

We have Eclipse installed and are all able to load code over to the Raspberry Pi. We started talking about the structure of the program that we are going to create and the requirements that drove us to the choices that we are making.

Most of what we are doing is the result of management decisions. Our playing field is going to be 12 feet by 12 feet. If you want to think of it as 4 meters by 4 meters or 8 cubits by 8 cubits, that works too. You are going to want to choose units for the project to make our computations easier to understand, both for us and the people who might look at our work later.

The field is going to have one or more obstacles placed on it, depending on how far we make it in our programming efforts. This means that you are going to have to work out a way to drive a certain distance, make an accurate turn, and still end up in the target area. The target is going to be a circle just large enough to contain the entire vehicle and the object of the exercise will be to get both wheels into the target area.

To achieve these goals, management has provided you the Shadow Chassis. Its wheel base is 16 centimeters from wheel center to wheel center and the wheels are 6.5 centimeters in diameter. They are driven by DAGU motor gearboxes. The motor gearboxes deliver 90 Revolutions Per Minute at maximum efficiency and 140 Revolutions Per Minute with no load. At the motor end of the construction there is a magnetic disc mounted, which will serve to give you the ability to determine how many times the motors have spun the wheels on their axis.

To read the magnetic discs, you are being provided with a digital input from a Hall Sensor (A Hall Sensor just senses electric currents. Any time a magnetic field moves, it generates current.) and a controller to drive the two wheels. In addition you will have a gyro to read the angular velocity of the vehicle and a distance sensor to tell you if you are about to run into anything.

Breaking that down, we are going to have to code modules to:

1. Integrate the angular velocity from the gyro
2. Integrate the ticks from the DAGU encoders
3. Integrate the distance readings from the distance sensor
4. Send speed and direction commands to both motors
5. Sum up the integrated inputs to determine where to vehicle is on the field
6. Plan how get from the current position to the target, missing the obstacle

The flow of control in the program will have to be such that, before we make the next move, we figure out where we are relative to the target and the obstacle, compute a move to get us closer to the goal while avoiding the obstacle, sending that move to the motors, and looping back to determine our current position.

We have a couple of processing decisions to make, and we are going to need some real data to make them. We are getting instantaneous data from our gyro and from our encoders. We want to get the best estimate we can for both of these devices. If we sample every second, we will be missing quite a few samples and our estimates will be off. Our first design consideration will be how often we have to sample our sensor data.

NOTE: We have some soldering to do. Management has provided us with gyros and motor controllers, but they do not have leads attached. We are going to have to get that done before too long. In the meantime, we can just run the motors at full speed by hooking them directly to our power supply. We have four batteries connected in series, which gives us 6 volts DC. The motors are nominally rated for 4.5 volts DC, But they can handle 6 volts DC without any issues.

There are a couple of ways we can approach this problem, but I think we are going to start with the encoders. If we plug the motor gearboxes directly in to our batteries, without the wheels connected, they are going to spin the output axle about 2 to 3 times a second. That should be a result of the motor shaft spinning about 3 \* 48 times a second, or about 144 times a second.

We are running the Raspberry Pi 3 this summer. The Pi 3 has a processor speed of 1.2 gigahertz. That means it has 1.2 x 10^9 cycles to handle all of the tasks it has to do. Between tick of the encoders, the Pi 3 can execute approximately 8 million instructions. This may seem like a lot, but your operating system has a lot of things going on. For each tick we are going to just want to increment the tick counter and make a note of the time that we performed the last increment, perhaps updating some statistical information, but not very much. When a program needs to check the values, they can be filtered and generally tweaked to provide the data that we need.

To accomplish this we are going to have to write a monitor to tell us when the voltage on the pin is rising or falling or has held high for a certain amount of time or has remained low for a certain amount of time. We are interested in capturing every transition of the pin from low to high, however we manage to do that.

Take a look at the DAGU encoder manual and the section on pin transitions in the pi4j manual and see what you can make of them.  
  
Wheel Encoder Kit: <https://www.sparkfun.com/products/12629>   
  
Pi4J Listener: <http://pi4j.com/example/listener.html>

Since our wheels are 6.5 centimeters in diameter, they will travel nearly 20.5 centimeters on each rotation. Use that to work out the distance per click and any other values you think we might need.

We will hash out the entire design of the encoder module on Wednesday and mount the motors and encoders on the base of the chassis. If we have time, we will solder some of the connections for the motor drivers and the gyros.