Golem Wing: Resurrection

Students: Ivan Dario Jimenez
Alex Gurney
CS3651
Date Submitted: May 5, 2016

I. Overview

The Golem Wing was originally invented by Saul Reynolds in collaboration with Mike Stilman. The idea was to have a balancing holonomic robot that would be able move in any orientation and direction in the plane as well as stay upright for future purposes of manipulation. After attempting to balance using an external computer for controls the project was abandoned and left to gather dust for years in the IRIM lab.

Today we present a revival of this project where we try to recreate and improve on their previous achievements with a modern embedded Linux system: the Raspberry Pi. On-boarding the computer paves the way for fully tether-free performance as well as a tighter control loop necessary for maintaining the robot balanced.

II. Hardware

A. Electrical

Once our robot had mounted motors it needed an embedded micro-controller to drive them and to read sensor values. The final solution would have to aggregate a gyroscope, an accelerometer and a send commands to three motors in series.

The RX-24F motors connect with a 4 pin connector that has a ground, source and an RS485 bus. On top of the bus the motors require a custom proprietary package format in order to communicate with the motors.

Although the option of using the Teensy was thoroughly investiaged, the team opted for going with a different alternative. Using the Teensy would have required us to debug a packet protocol and a bus protocol at the same time without much feedback to determine what is wrong other than an oscilloscope. Although possible, it was outside the scope of time granted to complete the project.

Fig. 1: Internal electrical connections of the current iteration of the Golem Wing. You can see from left to right the Dynamixel Dongle, the Dynamixel DC power sourcde and the raspberry PI. On the bottom you can see the current connection between the dongle, the motors and the power souce. For illustrative purposes you can see the Dynamixel Dongle's VC cable cut.

Instead we opted for a Raspberry Pi which could run a ROS node that we knew could communicate with the motors. We tried using a Raspberry 2 but the required package was incompatible with the latest version of raspbian compatible with that machine. Since installation on the Raspberry Pi 2 was impossible we moved to a Raspberry Pi 3 which did support the required OS.

This was using a proprietary USB Dongle that translated form a USB serial connection to RS485. Since the Dongle and Raspberry cannot supply enough current to move all three motors we had to purchase a separate power supply. Our final solution involved connecting the power supply's ground and power to the ground and power of the motors, and the ground of the dongle. We passed through the Bus from the dongle to the motors and cut the source that came from the dongle from the system. It was necessary to connect all grounds in order to ensure the that the BUS worked correctly.

With a suitable micro-controller we moved to connecting the desired sensor. We had to decide on the exact location since we had to choose between the top and bottom of the robot. We decided to to place the sensors as high on the robot as possible to increase sensitivity of the sensor. We worked at first with the ADXL345 accelerometer through the I2C bus.

Although The I2C bus was compatible with the Gyroscope 9-DoF stick solution we moved away from that because of a lack of libraries in the PI to read the values off the Gyroscope. This is because a Gyroscope needs to aggregate an accelerometer in order to avoid drift as well as translate the internal measurement to a meaningfull unit. Without a library to do those two things for us, completing the project in time would have been unrealistic.

Finally we opted for an Ardupilot that was pluggable into a USB port because there was a a library available that integrated well with our software stack.

B. Mechanical

III. Software

A. Previous Iterations

B. ROS Application Architecture

C. Controls

IV. Results

In the end our software and hardware stack managed to balance the robot albeit with some oscillation caused by the controller. Although we tried a wide range of parameters for our PID controller, the robot continued to oscillate even in the most stable configurations. We can attribute this to the exceedingly low center of mass of the robot. Even with a fast control loop the wheels are unable to keep it stable without oscillation.

We also performed stress tests on this control loop. Under some configurations we managed to obtain good recovery after perturbation of the system but it didn't always recover. The Golem Wing remains fairly unstable by design and thus an interesting control problem.

Fig. 2: Connection we built between the motor and the bearings.

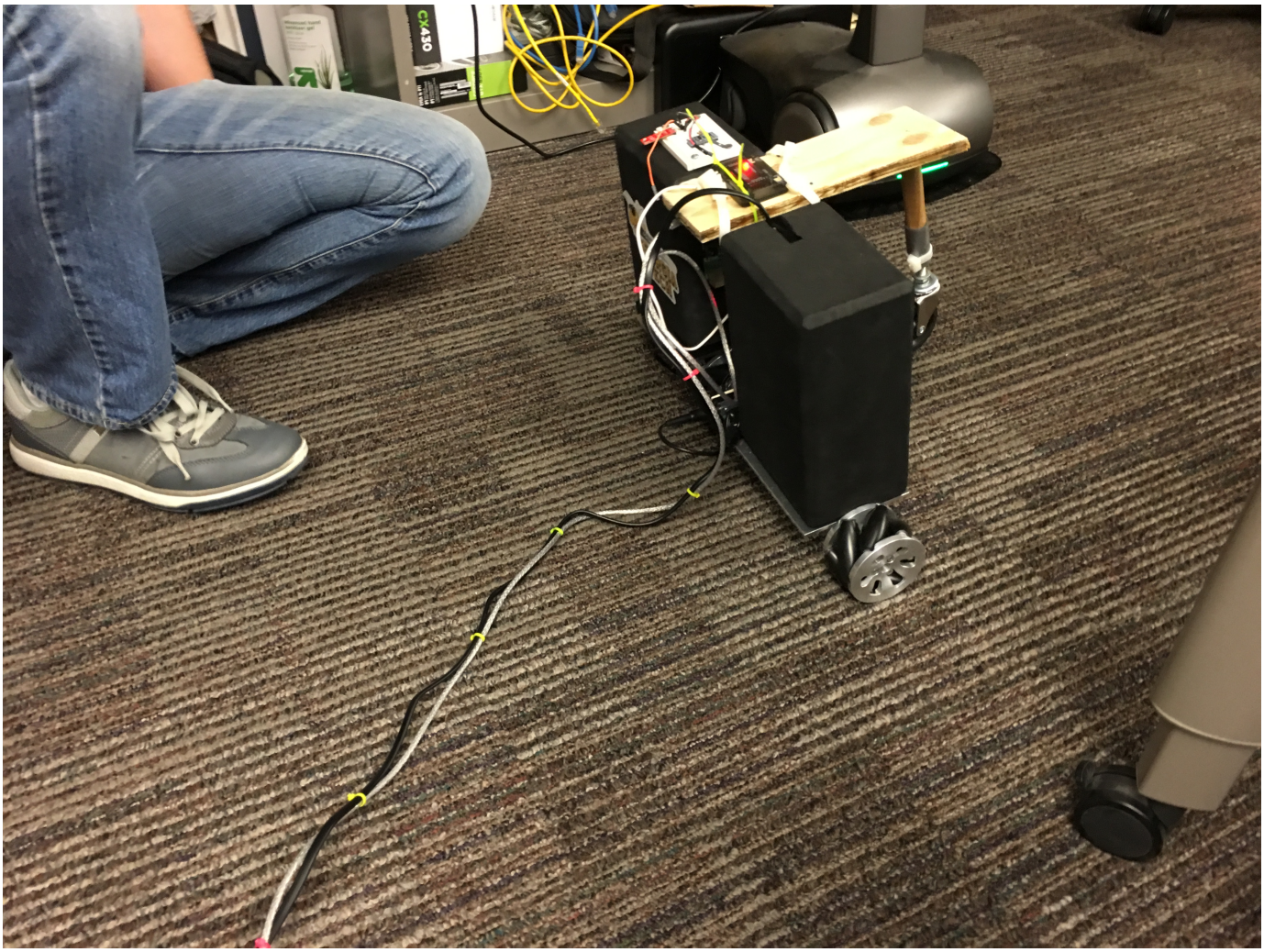

Fig. 3: The Balancing Golem Wing

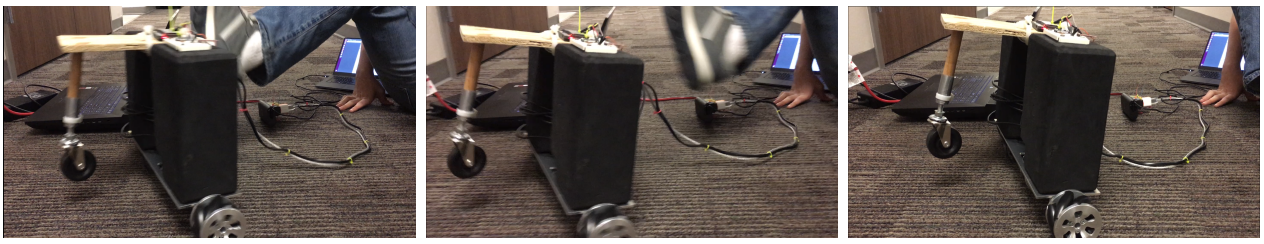

Fig. 4: Perturbation and recovery