IGVC: The University of Akron

Final Report

Design Team

Austin Beery

Brian Haynes

Tyler Wengerd

Faculty Advisor

Dr. Hartley

May 8th, 2011

Table of Contents

Mechanical_____page 1-4

- Motors
- Drive System
- Calculations
- Frame, sprockets, bearings
- Frame drawings

Electrical_____page 5-9

- Sensors, GPS
- Motor controller, motors
- Transmitter, receiver
- E-stop
- Schematic of device layout

Computer/Programming_____page 10-11

- Programming of components
- Sample code

IGVC Mechanical Drive System

In previous years, the mechanical drive system consisted of a power wheelchair base. This design proved to be insufficient because of the field conditions that robot was going through. The rough terrain and sometimes muddy conditions caused the robot to get stuck. We decided to go with a four wheel chain drive system. The reason for this choice was that it was a simple, practical choice that best suited our needs. While some consideration was given to do a track system, it would have been too expensive and complex and it would have been over designing.

Motors

Our entire design was based off the motors because we decided to recycle the motors from the old design. We did this because the motors were powerful enough and it was cost efficient.

Calculations to find torque for one motor:

Current Needed: 14 Amps Voltage Needed: 24 Volts Rotational Speed(w): 60 rpm

Power Produced: (Voltage)x(Amperage)= 336 Watts = 0.454 hp

Converting horsepower to ft-lbs: (5252*Hp)/(60)= 253.67 lb-ft

Four-Wheel Chain Drive System

Calculations to find max load force the drive system can move:

Assumption: Since the sprockets gears are same ratio, we are taking the 253.67 lb-ft to include the sprockets

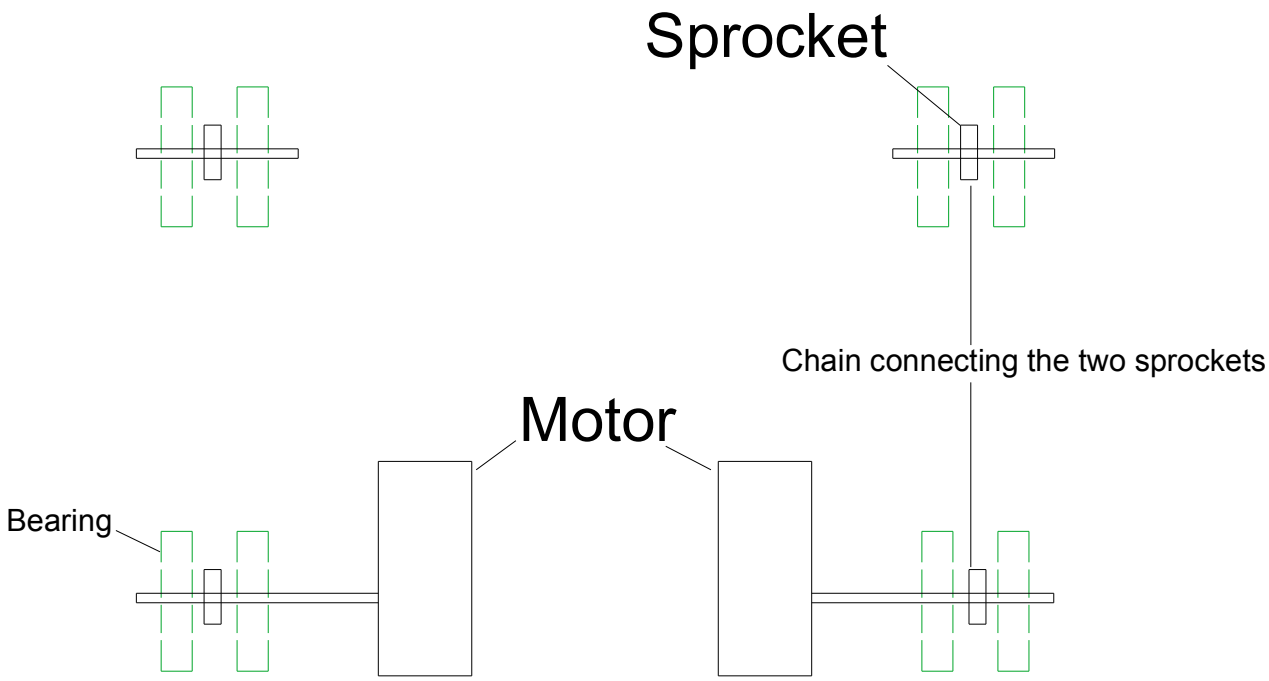
Radius of wheel: 0.583 ft

Max load force: T=(radius) x (Force) Max Force = 434.87 lbs.

Calculations for minimum load to maintain Traction on Dry Grass:

Friction Coefficient: 0.5

Friction Force = (Max load)x(Friction Coefficient) = 217.43 lbs.



Bearings

When looking for the bearings, we made to sure to find bearings that would be able to support a load of 430 pounds and fit a 0.75 inch shaft. While spacers were needed in order to line the bearings up with the shaft and numerous hours were spent in machining them, it was the cheapest option that still allowed them to be within design specifications.

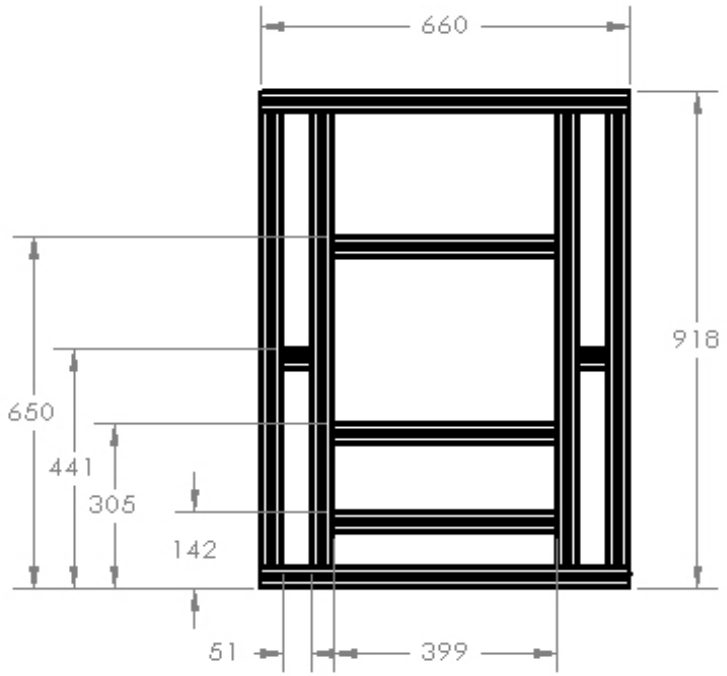
Sprockets

The sprockets have 16 teeth, with a 2 inch diameter. Four were used in order to obtain a one to one ratio.

Frame

The design requirments for the frame consisted of that would carry a payload, an aluminum box to house mechanical components, and batteries, which all together weigh approximately 60 pounds. We decided to use a ladder frame desgin because the simplicity

and the ability to hold the nececassary weight. Using extruded aluminum was weighed less then that of steel and of 20 pounds, plus it allowed for flexility in placing componets and the tensioning of the chain. The ease of assembly and cost also contributed to the decision to use aluminum.



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN MILIMETERS	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		NEXT PREFERRED TOLERANCING REF:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THE DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

IVGC BASE FRAME

SIZE	DWG. NO.	REV
A	Assem	

SCALE: 1:12	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

IGVC Electronic System

The previous design used a Lidar and more accurate GPS system. The components were quite high tech and hard to program. We used some of the components from the previous design. These items were the motors and Sabertooth motor controller. They helped with maintaining constant speed and torque. The electrical changes for this year's design will be mentioned in this section of the report.

The IGVC robot will use a basic stamp, sensors, GPS, and a speed controller to work properly. The sensors are ultrasonic and line sensors are used to help the robot know what is around the robot at all times, and the line sensors will help keep the robot in the lane of the figure eight course. By these components working properly, the robot will run the course to the best of its ability and try to finish the figure 8 course and reach the specified GPS destinations. These sensors are described in more detail in the next paragraphs.

The basic stamp runs on a 9 volt DC battery and sets pins high or low for the sensors to work properly. The ultrasonic sensors are strategically placed around the robot so the robot can find out what is around it. For instance, if an object is in front of the robot and there is an object to the left but not on the right then the robot will turn to the right and avoid the objects. These sensors seemed to work better than the lidar due to many sensors getting data rather than one beam of light that could only detect one object at a time. The sensors are programmed, as will be described in more detail in the programming report, to detect objects at fixed distances away from the robot and clear them accordingly. The GPS is precise and can get to the GPS point with 5 feet of error.

By placing the GPS in the middle of the robot, we ensure that we will be within the area. The last type of sensor is the line sensors. These sensors are placed close to the ground and can read the color of the lines to tell the robot that we are at the limit of the course lane and need to turn to keep within the course limits. These sensors are programmed to read a certain color and will let the robot know when that color is met.

The motor controller, that controls the motors of our robot, is programmed and runs with the basic stamp to make the motors turn on or off to make the robot move a certain direction that the basic stamp says. The motors run on 24 volts DC, 14 A minimum, and have a torque of 253.67 ft-lb to move the robot. The batteries we are using are Lithium Iron Phosphate batteries that will give us 20Ahr of run time.

Calculations: Ahr = current x time

$$20 = 14 \times t$$

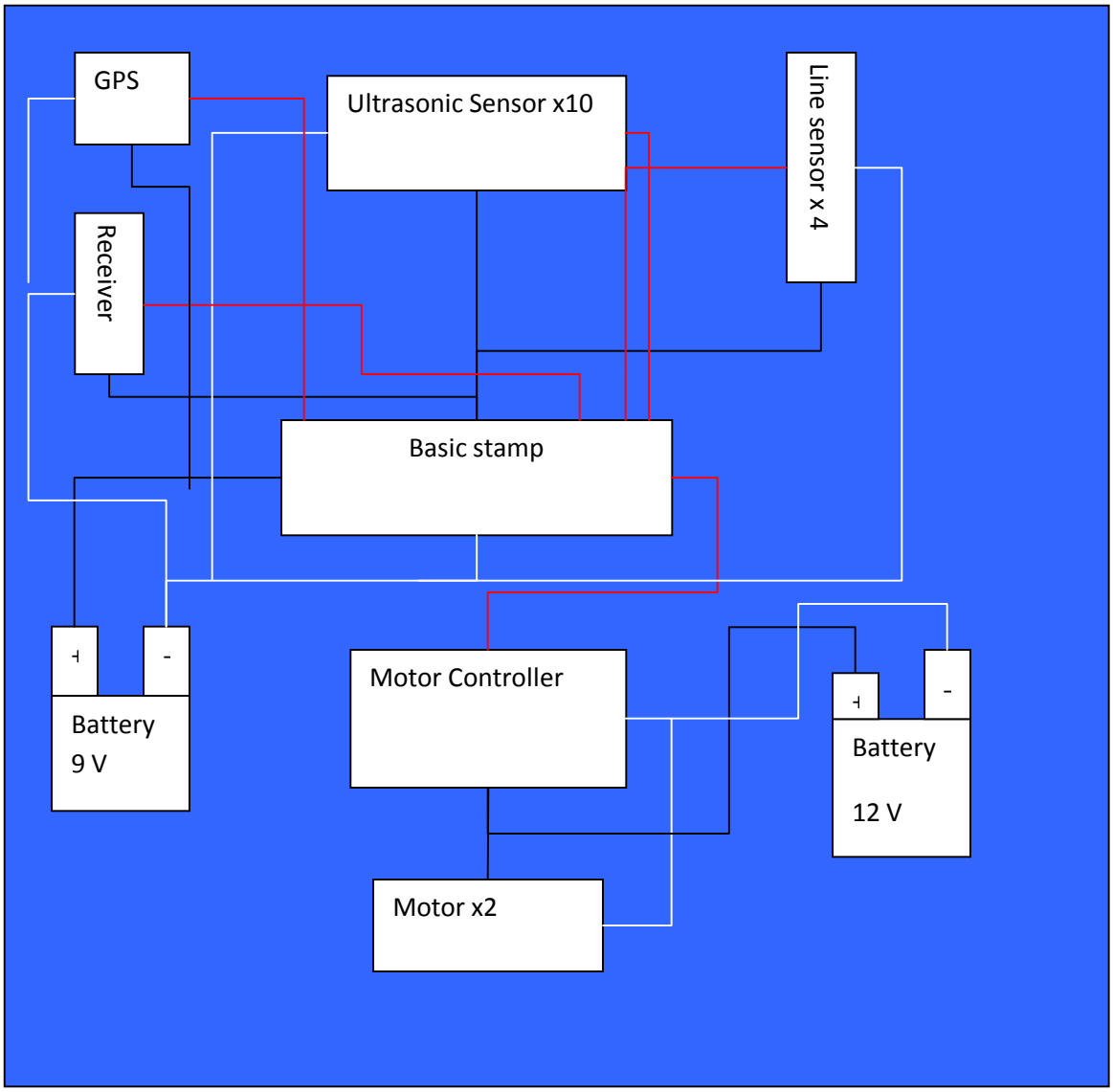
$$t = 1.43 \text{ hours}$$

We shortened our run time but decreased our weight a significant amount. These motors have their own set of batteries to avoid interference and to make sure that the other components do not make the battery run low too quickly. We chose these batteries due to them being light weight, consistent battery life, and easy charging.

Our robot will have two E-stops. One that is wired and the other is wireless. The one attached to the robot will be a push button and will short out the motor controller to make sure that the robot can not move once the button is pushed. The wireless one is programmed on the Basic Stamp. The receiver is on the stamp and once that gets a signal

from the transmitter that is in one of our teammate's hands then the stamp will stop all commands to the robot and shut it down. The transmitter and receiver are rated to work for the 100 ft limit for the wireless E-stop rules of the competition.

The electrical design is made to be simple for modular convenience. If something breaks down, then it should not take very long to find the problem and be able to replace the broken part. In Figure 2, the schematic is shown to show the circuit for how everything is put together to work properly.



Key: _____ Black lines = positive side of battery

White Line = negative side of battery

Red Lines = signal wires

IGVC Programming Outline

The IGVC robot is programmed using pBasic code on a BASIC stamp. It collects various data from the following sensors:

- Line sensor
    - Gives a certain value if a line is detected, i.e., a certain color is 'seen'
- PING ultrasonic sensor
    - Returns distance to any object found
- Wireless receiver
    - For the emergency stop
- GPS sensor
    - Returns time, latitude, longitude, altitude, speed, and travel direction/heading,
- Motor controller
    - Different speeds on the right and left motor controllers will allow turning and driving forward and reverse.

The robot is programmed to collect data from the GPS sensor to find the distance of a desired latitude and longitude. It then calculates the longest distance it can possibly go in the most efficient direction (the direction that takes the robot closer to the goal). Following that, it follows routines to drive forward or backward, as well as turn, in an effort to reach the destination. This then loops until the goal is reached.

All the while, the robot checks the emergency stop status to see if a halt is needed. The e-stop kills all routines. For the e-stop, there is a physical kill switch as well as a wireless receiver which can stop all activity on the robot from a distance of at 100 feet.

A rough outline of the algorithms used follows.

```
//initial variables
desiredLocation;
currentSpeed;
currentLocation;
eStopOn; //boolean, is emergency stop sensor on
arrived; //have we arrived at the destination
distanceToDestination
```

```
if(no final destination set)
        //find final destination
        get desired gps coordinates
        calculate distance

//Check e-stop and arrived status
routine:checkstop
if eStop activated
        stop completely
if arrived (gps coordinates = desired coordinates +/- 1 foot
        stop, give arrived status


//find current location
routine:getlocation
        run estop routine
        save current heading
        get current gps coordinates
        get current speed


//find path
routine:findpath
        subroutine: check stop
        look forward for a path towards destination
        calculate longest path and direction given by front three sensors and recent line
sensor activation
        if direction is straight ahead, no turning needed
                subroutine: drive
                check for arrivsl
        if path is not straight ahead and direction is valid
                subroutine: turning
                subroutine: drive
                check for arrival
        if direction from three from sensors is not a valid direction (due to object, line
sensor)
                find best path given by rear line sensors
                subroutine: turning
                subroutine: reverse driving
                check for arrival

// turning
routine:turning
        from desired heading vs current heading, calculate angle to turn
        if turn left is needed
                for count of degrees needed to turn
```

```
                    run motors at different speeds and directions to achieve a degree of
turn
                    run estop check
                    if line sensor activated
                        stop turn, reverse turn slightly, rerun find path
                    increment degree counter (debug)


    if turn right is needed
        for count of degrees needed to turn
                    run motors at different speeds and directions to achieve a degree of
turn
                    run estop check
                    if line sensor activated
                        stop turn, reverse turn slightly, rerun find path
                    increment degree counter (debug)


routine: driving
    for count of inches/centimeters desired to travel - 1 foot (unless distance < 1 foot)
        run estop check
        run motors to move forward
        if line sensor activated
            stop, back up a little if possible, rerun find path
        increment distance counter (debug)


routine: reverse
    for count of inches/centimeters desired to travel - 1 foot (unless distance < 1 foot)
        run estop check
        run motors to move backward
        if line sensor activated
            stop, go opposite direction up a little if possible, rerun find path
        increment distance counter (debug)
        check if arrived
```