EEL 4665L

Intelligent Machine Design Lab

Spring 2012

Final Report

4/23/12

LandYachtBot

Michael Feliciano

Instructors

Dr. Arroyo

Dr. Schwartz

TAs

Ryan Stevens

Tim Martin

Josh Weaver

Table of Contents

I. Abstract

LandYachtBot is an autonomous land sailing vehicle. Its primary objective is to follow a set of GPS waypoints while using the wind as its sole means of locomotion. Additionally, the yacht can avoid obstacles either by attempting to turn away from them, or by deploying a mechanical brake if it senses an imminent collision with an object. LandYachtBot detects objects in its path by using two front-mounted sonar modules. To sail autonomously, the yacht uses a variety of sensors including a wind vane, a GPS, and a compass.

II. Executive Summary

LandYachtBot was originally conceived as a GPS waypoint following, autonomous sailboat. I have always been interested in sailing and thought that a robot powered entirely by the wind would be a unique and challenging project for IMDL. I was correct in that assumption.

After deciding that the project would be further complicated by the presence of water, I decided that the sailing algorithms could be implemented on a land yacht for less cost, easier debugging, etc. Land yachts also deal with much less friction than sailboats (as water is much thicker than air, and the wheels negate friction even further) allowing them to be driven to higher speeds in less wind.

Controlling a small autonomous sailing vehicle requires advanced control algorithms, a number of different sensors, and several powerful servos. The main control unit on LandYachtBot is the Epiphany DIY board which houses a powerful ATxmega61A microcontroller. The control algorithms and sensor data are processed and executed using this chip.

The sensor suite used to gather the necessary data to make decisions on how to sail include a Locosys LS20031 GPS antenna, a wind vane built using a MA3 shaft encoder, a MicroMag3 magnetometer, and two LV-MaxSonar-EZ4 sonar modules for obstacle avoidance.

The sonar modules and the wind vane encoder (as well as a potentiometer at the base of the mast) provide analog data to the Xmega. The magnetometer (or compass sensor) communicates with the Xmega via SPI. Finally the LS20031 communicates with the Xmega via a TTL serial connection.

III. Introduction

A land yacht is a three-wheeled vehicle that is powered by wind. The basic principles of sailing in a boat on water apply to a land yacht on a smooth surface. Instead of a rudder and keel, a land yacht relies on a steering wheel and low center of gravity.

To sail with the wind or perpendicular to it is rather rudimentary. To sail into the wind requires the yacht to perform a series of maneuvers called “tacking.” Tacking means to sail at a 45 degree

angle to the incoming wind, then quickly cut across 90 degrees such that the yacht is now sailing at a 45 degree angle to the wind on the other side. Doing this repeatedly, the yacht can sail into the wind, albeit while traveling twice or more the distance than it displaces.

Performing autonomous sailing and navigation requires advanced control algorithms, as well as highly responsive mechanical systems. Using a powerful Epiphany DIY microcontroller board, an array of sensors, and several special servos, LandYachtBot is equipped with the means to have both.

IV. Integrated System

The brain of LandYachtBot is the Epiphany DIY board which houses a powerful Atmel Xmega microcontroller and numerous servo controllers. All sensor data is read into the Xmega processor and decisions are made based on a variety of factors. Once the decision is made, the servo controller will move one or more of four servos to their required positions.

In general, LandYachtBot will attempt to first follow its GPS waypoints. If the wind direction is not favorable or an obstacle is detected, LandYachtBot will then try to compensate (by tacking or turning away, respectively).

V. Mobile Platform

LandYachtBot‘s mechanical platform consists of a T-shaped wooden frame with a thin, wooden “deck” underneath to house the electronic components. The frame is approximately 30 inches long and approximately 20 inches wide at the rear. The mast is about 4 feet tall with a nylon triangular sail.

The wheels are 4 inch Razor® scooter wheels. Two wheels are fixed in the back and the front wheel is attached to a steering servo. The servos themselves will be placed into cut outs in the frame.

VI. Actuation

There are no drive motors on LandYachtBot. It relies exclusively on the wind (or a very powerful fan or blower) for locomotion.

LandYachtBot has four servos. The steering servo is a robust quarter-scale standard analog servo. This servo needs to provide enough torque that high wind will not buckle the steering system. The sail (or, more correctly, the boom) will be controlled using a quarter-scale “sail arm” servo. These powerful servos have a much longer arm than standard servos for greater control range. The tension on the sail will be controlled with a winch servo, which can make several full rotations with the same control signal. This is important because the speed of a land yacht can be

regulated in part by slacking or tightening the sail. Finally the braking servo will be a standard size servo which can raise and lower a brake the few inches from the frame to the ground.

VII. Sensors

LandYachtBot will have several sensors which, for convenience I will classify into two categories—conventional and unconventional.

Conventional sensors are sensors that do not fall in the "special sensor" category as defined in the class syllabus. LandYachtBot's two sonars, accelerometer, and compass fall in this category.

The sonar module I have selected is the LV-MaxSonar-EZ4. Two of these modules will be attached to the front of the yacht at an angle of approximately 20° from center. Because of the conical pattern of the emitted sound waves, I am confident that this configuration will be more than adequate for my obstacle detection. Initially I thought that one of these units would be sufficient because my obstacle avoidance system would have relied on a braking system, rather than steering. After testing this single sonar set up, it quickly became apparent that two sonars are needed to perform obstacle avoidance efficiently.

Fig. 1 LV-MaxSonar-EZ4

The 3-axis accelerometer module I chose is an ADXL335 accelerometer with breakout board. This accelerometer gives an analog output and is rated to ±3g which is good enough for my purposes. This sensor, in conjunction with my compass, was to form the basis for an inertial navigation system, which would have acted as a redundancy to my GPS navigation. Such functionality was not necessary and at the time of this report, the accelerometer is not being used.

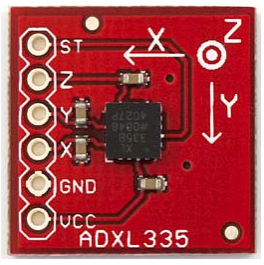

Fig. 2: ADXL335 3-axis accelerometer breakout board.

The compass I chose is a MicroMag3 3-axis magnetometer. This module uses a SPI interface to send data to the ATxmega61A on the Epiphany board. This sensor is important because it gives

the yacht its heading with respect to the wind direction when data is combined with data from the wind vane.

Fig. 3: MicroMag3 3-axis magnetometer.

The two most critical sensors I classify to be unconventional sensors, i.e. rarely used in this class. These sensors are my GPS antenna and my wind vane.

My GPS module is a LOCOSYS LS20031 66-channel GPS antenna module. While the yacht is operating outside, this module fixes the yachts current position and helps it determine where the next waypoint is.

Fig. 4: LS20031 GPS antenna module.

Finally my last sensor is my most important—the wind vane. I decided to use a low-friction ball-bearing analog MA3 magnetic shaft encoder from US Digital. After adding a simple vane to the shaft and adjusting the sensor to the proper angle with respect to the yacht, I have a reliable wind direction reading which is paramount to autonomous sailing.

Fig. 5: MA3 Miniature Absolute Magnetic Shaft Encoder

VIII. Behaviors

LandYachtBot's primary function is to navigate a set of GPS waypoints using only the wind for propulsion. Because the frame is heavier than I originally intended, the amount of wind required for sailing is more than most Gainesville days provide. Because of this, I have used powerful fans and leaf blowers as artificial wind to propel the yacht.

Sailing outdoors requires a large, flat, relatively smooth area. Because of this, the yacht has a separate indoor functionality in which the yacht rests on a turntable near a fan and demonstrates tacking and sailing downwind by steering, moving the sail, and adjusting sail tension appropriately. The turntable allows a human operator to simulate turning while the yacht is stationary.

IX. Experimental Layout and Results

The results of my sonar testing revealed that a single sonar was not adequate for obstacle avoidance. Adding a second sonar module was the right fix for the problem and now when an obstacle is detected, the steering performs so well that the brake is almost unnecessary as it is so infrequently used.

Testing my wind vane has proven to be more difficult as the sensor has to be manually adjusted to the correct angle every time the robot is assembled. Once calibrated, the sensor gives very good resolution even while operating at 3.3V instead of the recommended 5V.

By far my biggest problems have come from my GPS and compass sensors. I would have started testing on these sensors before the semester started if I had the opportunity to go back.

X. Conclusion

I set out to build LandYachtBot because it was unique. A robot like this had never been attempted before in this class. Along with its uniqueness came several unique challenges. The control algorithms for an autonomous sailing robot are highly complex, even after all of your sensors work as they should. I have and will continue to work to improve upon the foundation that I have laid until it is time for my final presentation. To be honest, I believe I have bitten off more than I can chew with this project. Nevertheless, I have enjoyed the challenges and frustrations that have gotten me to this point. If I were to take the class again knowing what I know now, I would choose to build LandYachtBot again without hesitation.

XI. Documentation

http://www.softwareresearch.net/fileadmin/src/docs/teaching/SS11/SaI/AutonomousSailingBoats_Wittinghofer_Alt.pdf

XII. Appendix

```
/*
** Epiphany DIY Founder:      Tim Martin
**
** Class:                           IMDL Spring 2012
** Professors:                      Drs. Arroyo and Schwartz
**
** Student:                         Michael Feliciano
** Robot:                           LandYachtBot
*/

#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"

#define DbLedOn()       (PORTR.OUTCLR = 0x02)             //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()             (PORTR.OUTSET = 0x02)             //Turns the debug
led off.  The led is connected with inverted logic
#define DbLedToggle()   (PORTR.OUTTGL = 0x02)            //Toggles the debug led off.
The led is connected with inverted logic

uint16_t sailingDownwind(uint16_t i);
uint16_t sailingUpwind(uint16_t j);
double windDirection();

int main (void)
{
   board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                              motorInit() is declared within because by default you
the user should define what your
                              motor setup is to prevent hurting the Epiphany.  You
can do this by */
   RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
   DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
   ATtinyServoInit();//this function initializes the servo controller on the Attiny
   uartInit(&USARTC0,115200);//as can be seen in the schematic.  This uart is
connected to the USB port.  This function initializes this uart
   uartInit(&USARTE1,9600);//as can be seen in the schematic.  This uart is connected
to the Xbee port.  This function initializes this uart
   ADCsInits();//this function initializes the ADCs inside the Xmega
```

```
    sei();
    LCDInit();

    uint16_t i = 0;
    uint16_t j = 0;
    uint16_t count = 0;

    setServoAngle(21,90);                    //set winch at half position

    setServoAngle(20,95);                    //sail arm neutral
    setServoAngle(8,85);                  //brake neutral
    setServoAngle(1,92);                  //steering neutral

    DbLedOff();

    fprintf(&lcd_str, "\rReady");
    _delay_ms(2000);

  while(1)
  {
    //Go straight
    if(analogRead_ADCA(0) >= 500 && analogRead_ADCA(1) >= 500)
    {
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",0,analogRead_ADCA(0));
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",1,analogRead_ADCA(1));
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",2,analogRead_ADCA(2));
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",3,analogRead_ADCA(3));
          DbLedOff();
          setServoAngle(1,92);
          setServoAngle(8,85);
          _delay_ms(50);
          fprintf(&lcd_str, "\rClear");

          if (i <= 7)
          {
                count = sailingDownwind(i);
                i = count;
          }
          else if (j <= 7)
          {
                count = sailingUpwind(j);
                j = count;

          }
          else
          {
                i = j = 7;
          }
    }
    //Turn right
    else if (analogRead_ADCA(0) < 500 && analogRead_ADCA(1) >= 500)
    {
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",0,analogRead_ADCA(0));
          //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",1,analogRead_ADCA(1));
          DbLedOn();
          fprintf(&lcd_str, "\rObstacle\nLeft");
```

```
            while(analogRead_ADCA(0) < 500 && analogRead_ADCA(1) >= 500)
            {
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",0,analogRead_ADCA(0));
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",1,analogRead_ADCA(1));
                    DbLedOn();
                    setServoAngle(1,135);
                    _delay_ms(300);
            }
            if(analogRead_ADCA(0) < 500 && analogRead_ADCA(1) < 500)
            {
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",0,analogRead_ADCA(0));
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",1,analogRead_ADCA(1));
                    DbLedOn();
                    setServoAngle(8,50);
                    fprintf(&lcd_str, "\rBREAK!!!");
            }
            setServoAngle(1,75);
            _delay_ms(300);
            setServoAngle(1,92);
    }
    //Turn left
    else if (analogRead_ADCA(0) >= 500 && analogRead_ADCA(1) < 500)
    {
            //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",0,analogRead_ADCA(0));
            //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",1,analogRead_ADCA(1));
            DbLedOn();
            fprintf(&lcd_str, "\rObstacle\nRight");
            while(analogRead_ADCA(0) >= 500 && analogRead_ADCA(1) < 500)
            {
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",0,analogRead_ADCA(0));
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",1,analogRead_ADCA(1));
                    DbLedOn();
                    setServoAngle(1,60);
                    _delay_ms(300);
            }
            if(analogRead_ADCA(0) < 500 && analogRead_ADCA(1) < 500)
            {
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",0,analogRead_ADCA(0));
                    //fprintf(&USB_str,"ADC channel %d
= %d\r\n\r\n",1,analogRead_ADCA(1));
                    DbLedOn();
                    setServoAngle(8,50);
                    fprintf(&lcd_str, "\rBREAK!!!");
            }
            setServoAngle(1,120);
            _delay_ms(300);
            setServoAngle(1,92);
    }
    //Deploy brake if both sonars detect an obstacle in front
```

```
    else
    {
            //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",0,analogRead_ADCA(0));
            //fprintf(&USB_str,"ADC channel %d = %d\r\n\r\n",1,analogRead_ADCA(1));
            DbLedOn();
            setServoAngle(8,50);
            fprintf(&lcd_str, "\rBREAK!!!");
    }

}
}

uint16_t sailingDownwind(uint16_t i)
{
    double windDir;
    windDir = windDirection();

    //initial maneuver, veer left
    if (i == 0)
    {
            while (windDir > 1100 && windDir < 1400)
            {
                    setServoAngle(1,60);
                    setServoAngle(20,30);
                    _delay_ms(100);
                    windDir = windDirection();
            }
    setServoAngle(1,120);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    //come about right
    else if (i == 1 || i == 3 || i == 5)
    {
            while (windDir >= 800)
            {
                    setServoAngle(1,135);
                    setServoAngle(20,150);
                    _delay_ms(100);
                    windDir = windDirection();
            }
    setServoAngle(1,75);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    //come about left
    else if (i == 2 || i == 4 || i == 6)
    {
            while (windDir <= 1500)
            {
                    setServoAngle(1,60);
                    setServoAngle(20,30);
                    _delay_ms(100);
                    windDir = windDirection();
```

```
        }
    setServoAngle(1,120);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    //turn around, veer right, head straight downwind
    else if (i == 7)
    {
        while (windDir >= 600)
        {
            setServoAngle(1,135);
            setServoAngle(20,150);
            _delay_ms(100);
            windDir = windDirection();
        }
    setServoAngle(1,75);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    i++;
    return i;
}

uint16_t sailingUpwind(uint16_t j)
{
    double windDir;
    windDir = windDirection();

    //initial maneuver, tack right
    if (j == 0 || j == 2 || j == 4 || j == 6)
    {
        while (windDir <= 600 && windDir >= 2100)
        {
            setServoAngle(1,135);
            setServoAngle(20,30);
            _delay_ms(100);
            windDir = windDirection();
        }
    setServoAngle(1,75);
    setServoAngle(20,150);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    //tack left
    else if (j == 1 || j == 3 || j == 5)
    {
        while (windDir <= 600)
        {
            setServoAngle(1,60);
            setServoAngle(20,150);
            _delay_ms(100);
            windDir = windDirection();
        }
```

```
    setServoAngle(1,120);
    setServoAngle(20,30);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    //turn around, veer right
    else if (j == 7)
    {
           while (windDir <= 1100 && windDir >= 1400)
           {
                  setServoAngle(1,135);
                  setServoAngle(20,30);
                  _delay_ms(100);
                  windDir = windDirection();
           }
    setServoAngle(1,75);
    _delay_ms(300);
    setServoAngle(1,92);
    _delay_ms(200);
    }
    j++;
    return j;
}

double windDirection()
{
    unsigned int wd1, wd2, wd3, wd4, wd5 = 0;
    double avgWindDir;

    wd1 = analogRead_ADCA(2);
    if (wd1 <= 400 || wd1 >= 2300)
    {
           wd1 = 0;
    }
    _delay_ms(100);
    wd2 = analogRead_ADCA(2);
    if (wd2 <= 400 || wd2 >= 2300)
    {
           wd2 = 0;
    }
    _delay_ms(100);
    wd3 = analogRead_ADCA(2);
    if (wd3 <= 400 || wd3 >= 2300)
    {
           wd3 = 0;
    }
    _delay_ms(100);
    wd4 = analogRead_ADCA(2);
    if (wd4 <= 400 || wd4 >= 2300)
    {
           wd4 = 0;
    }
    _delay_ms(100);
    wd5 = analogRead_ADCA(2);
    if (wd5 <= 400 || wd5 >= 2300)
```

```
        {
                wd5 = 0;
        }
        _delay_ms(100);

        avgWindDir = (wd1 + wd2 + wd3 + wd4 + wd5) / 5.0;

        return avgWindDir;
    }
```