Implementation Notes: Dead Reckoning Robot

Senior Design
Spring 2003
Texas A&M University

Lee Fithian
Ajay Joseph
Steven Parkinson
Saba Rizvi

1 Introduction

This section describes a description of the intended audience, the purpose of this document and a description of the topics covered.

1.1 Audience

This document is designed to present the reasoning behind the design used for the Dead Reckoning Robot project. The level of detail presented in this document is much more through than that of the Users Manual document.

The increased detail is to spare future engineers and scientist working on the project from the burden of reverse engineering the project when making modifications or adding to the design.

1.2 Section Descriptions

This section describes the purpose of each of the sections contained within this document. Following the Introduction is a description of why the Mark III robot with an OOPic microcontroller was used as the basis of the project.

Following the description of the Mark III robot is a section with information about each of the sensors. The section for each sensor includes the following subsections:

- Device Information
    - Description of the manufacture of the sensor.
    - Resources for information about the sensor.
- OOPic Interface
    - Description of the physical interface to the OOPic microcontroller board.
    - Description of the details of the software component of the interface.
- Issues
    - Description of issues encountered while interfacing the sensor.

2 Mark III Complete Kit (OOPic Version)

This section describes reasons the Mark III robot and OOPic microprocessor were chosen. Also included in this section is a description of the modifications made to the chassis to facilitate the addition of the sensors.

2.1 OOPic Microcontroller

The OOPic microcontroller was chosen for several of its convenient features. These features include:

- Simple to Program.
- Wide variety of built in functions.
- Quantity of I/O ports.
- Cost.

The OOPic microcontroller is programmed using a subset of three different popular programming languages. The languages available are Visual Basic, Java and C. Visual Basic is the language used in this project. The ability to quickly create programs using object oriented techniques is the main factor contributing to the use of Visual Basic.

Because the OOPic microcontroller is a mature product there is support within the OOPic libraries for a wide variety of hardware available for robotics. By utilizing support within the OOPic the design team greatly decreased the code development time required.

The OOPic also has advantages over many other products available because it has a large number of I/O ports arranged on a standard 40 pin header. Seven of these ports are available for sampling analog values using the A/D converted built into the OOPic.

The low cost of the OOPic board contributed to the selection of this microcontroller.

2.2 Mark III Robot

The Mark III Robot is used as the basis of the Dead Reckoning Robot. The Mark III robot is available in a complete kit which includes:

- Servos, modified for continuous motion as the source of propulsion.
- Electronics, including the OOPic microcontroller.
- Chassis
- Wheels

The Mark III is easy to modify to meet the requirements of the design. By using the Mark III kit the design team avoided sinking a large amount of time into the development of robot, many varieties of which are commercially available. Instead the time was used to develop the interface to the sensors and developing code to utilize the sensors.

2.3 Chassis Modifications

There are several chassis modifications necessary to integrate the four sensors. The first modification discussed is the addition of a second PCB (with the first PCB being the OOPic microcontroller PCB). The second modification discussed in this section details how the shaft encoders are attached to the chassis of the robot and to the encoders.

2.3.1 Sensor PCB

To attach the various sensors and circuit elements to support the sensors a low cost prototyping PCB with a pad per hole PCB pattern is used. A four by six inch board allows plenty of space for all sensors and facilitates future expansion.

Four pieces of threaded rod cut to a length of approximately two and one half inches are used to attach the PCB to the chassis. The threaded rod pieces are threaded into the end of the posts which the OOPic microcontroller PCB rests upon. A nut is threaded onto the rods to hold the microcontroller PCB in place.

Nuts are threaded onto the four posts to hold the PCB in place. One more nut per rod is threaded approximately on half an inch onto the rods. The second PCB rests on this

2.3.2 Encoder Attachments

An axle is extended from each of the servos. Additionally, the housing of the encoder is secured to the chassis of the robot.

To accomplish this a special machine screw is used. The machine screw fits into the existing threads of the servo wheel attachment and is long enough that it extends beyond the wheel and through the shaft encoder.

Because the diameter of the hole in the shaft encoder is larger than the diameter of the machine screw a sleeve is required to mount the encoder to the machine screw. A sleeve with a outer diameter slightly less than the diameter of the shaft encoder and a threaded hole with diameter equal to that of the machine screw is used to attach the machine screw to the shaft encoder. The end of the screw is screwed into the screw hole in the servo as the encoder is attached to the chassis.

The housing of the encoders are attached to the chassis of the Mark III robot using two custom L-brackets. The L-brackets are constructed of 22 gauge weld steel sheet metal. Pieces of the metal are cut ½ inch wide and approximately 4 inches long. The length of the piece is dependent upon the length of the machine screw selected. Four of these pieces are cut.

The pieces cut are then bent into a ‘L’ shape. Holes are drilled into the metal corresponding to the holes in the chassis. The holes referred used on the chassis were the two which attached the scoop to the main part of the chassis and the on the lip of the main part of the body opposite to the scoop.

It is recommended that the holes drilled in the L-brackets which attach to the chassis be made wide so that the distance the encoder is from the wheel can be easily adjusted when attaching the encoders.

Care must be taken such that when the encoders are attached the code wheel does not contact any surfaces within the encoder. This damages the code wheel. If this situation exists it will often cause a ‘squeaking’ noise when the encoders are turning.

2.3.3 40 Pin Cable

To avoid an unsightly and unmanageable number of wires from the bottom of the upper PCB to the OOPic forty pin interface a standard forty pin IDE header is soldered to the upper PCB. An IDE cable is cut to the necessary length and connects the forty pin header on the OOPic to the forty pin header on the upper PCB. The wires from each of the sensors go to the forty pin header on the upper PCB.

When it is necessary to remove the upper PCB the IDE cable can simply be disconnected. This eliminates the needs to wrap all the wires to the forty pin header on the OOPic board each time the upper PCB is removed.

2.3.4 Power Supply

During testing the rate of power drain prevented the design team from working for the time periods necessary. A 9 V, 300 mA power supply is used to decrease the dependency on battery available and recharge rates.

3 Accelerometer

This section provides information about the accelerometer which was chosen for the design of the Dead Reckoning Robot. The subsections below contain information about the device itself; hardware and software details of the OOPic interface and issues that arose during integration of the sensor.

3.1 Device Information

The ADXL202E was the accelerometer that was chosen to be used in this dead reckoning robot. It is a dual axis accelerometer which has a measurement range of +/- 2g.

The acceleration can be read in 2 different ways. The first way uses Duty Cycle Modulated signals in which a pulse T1, which measures the acceleration is compared to the total period of the whole signal, T2. A ratio of 50% relates to zero acceleration in that axis. In either the positive or negative direction, a 12.5% step from the center (50%) would correspond to +/- 1 g, respectively.

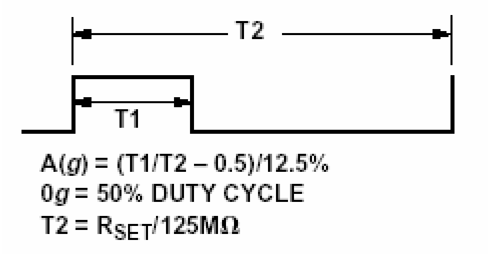


Figure 1: Accelerometer DCM Output

The second way the accelerometer can be read is in analog mode. An analog to digital converter would be needed for each axes output.

- ADXL202E Accelerometer Product Page
    - http://www.analog.com/Analog_Root/productPage/productHome/0,2121,generic%253DADXL202%2526level4%253D%25252D1%2526level1%253D212%2526level2%253D213%2526level3%253D%25252D1%2526resourceWebLawID%253D0,00.html

3.2 OOPic Interface

This section describes the details of the interface to the OOPic microcontroller.

3.2.1 Physical Interface

This section provides the details of the physical electronic interface for this sensor.

3.2.1.1 Physical Location

The accelerometer was chosen to be placed at the center of rotation on the robot. This was to help achieve better acceleration readings in both the X and Y axes. This required the gyroscope to be offset from the center by a slight margin.

3.2.1.2 Socket Mount and Tuning Circuit

Since the accelerometer is a 5 mm x 5 mm x 2 mm 8-lead hermetic LCC package, it required a socket so that it could be mounted on the sensor PCB. After a long search process, a socket was found for the particular size LCC. Unfortunately, the pins for the socket were not in a standard PCB layout. Only the inner four pins complied to a standard layout while the outer four pins would not line up. To combat this issue, slots were drilled so that the outer four pins would be able to protrude through the PCB. Wires would then be wrapped on the outer pins while the four inner pins were soldered to the PCB directly.

The wires from the accelerometer socket were then connected to a 20 pin wire wrapping IC socket. The IC socket was used to hold various resistors and capacitors so that the accelerometer's bandwidth could be set and the outputs could be filtered. This was done with future considerations in mind so that bandwidth can be changed easily by pulling out the current resistor and changing it with the new one.

3.2.1.3 Circuit Diagram

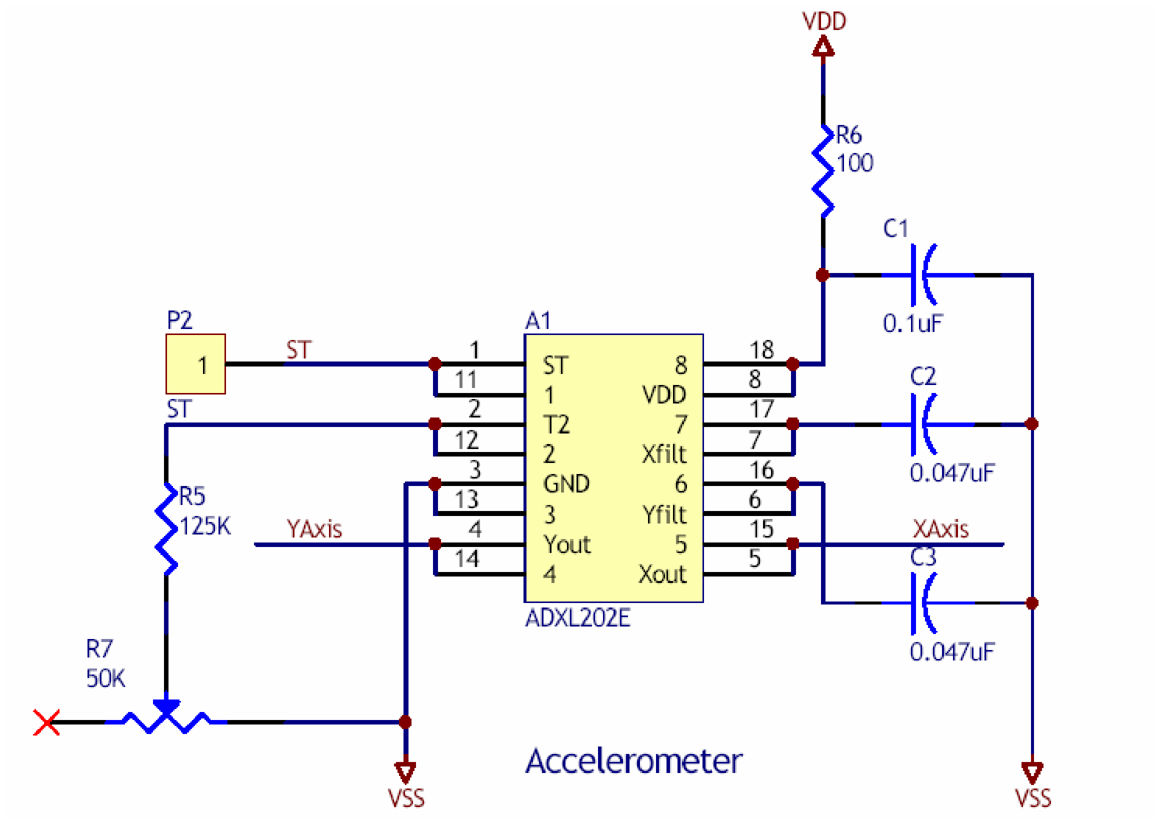


Figure 2: Accelerometer Circuit Schematic

Slight modifications were made to this schematic. R5 and R7 were replaced with a 1.25 MΩ resistor.

3.2.1.4 Pinout

The following table describes how the accelerometer was connected to our 40 pin OOPic connector.

Note that the OOPic **PIN** is not the same number as the OOPic I/O line.

Accelerometer Pin	Description	OOPic Pin
3	GND	24
6	Y Filter (Y Output Analog)	9
7	X Filter (X Output Analog)	13
8	+5 VDD	22

3.2.2 Software Interface

In order to use the use the digital output of the accelerometer, a fast enough counter must be used. Unfortunately the OOPic's counter was not fast enough to accommodate the accelerometer's output signal. Another method using another chip, the PAK VIIa, was attempted, but did not work out (see issues). The analog outputs were then used for the measurements.

The following code shows how to instantiate the A/D converter object necessary to read the outputs of the accelerometer. The oA2D10 object is part of the OOPic library.

```
Dim xvolt As New oA2D10

Sub Main()
  distTrav = 0
  totalRates = 0
  positiveRate = 0
  cntNew = 0

  do
    currentRate = xvolt.value
    if (currentRate > zeroRateX) then
       positiveRate = positiveRate + 1
      totalRates = totalRates + currentRate
      avgRate = totalRates / positiveRate
    end if

    vel = (((((avgRate - zeroRateX) * 980) / 63) * cntNew) / 283)
    distTrav = distTrav + ((vel * cntNew * k) / 283)
    cntNew = count.value - cntNew
  loop while (distTrav < (x * k))
End Sub
```

This code segment records all acceleration values and keeps averaging them so that a velocity can be found. This velocity is then integrated to achieve the distance traveled. This is achieved with the aid of a virtual circuit clock that is always running. Time can measured with a simple conversion of count ticks to seconds.

3.3 Issues

This section describes problems the design team encountered.

3.3.1 Socket Mount

This was mentioned in the physical interface section.

3.3.2 PAK VIIa

In order to try and use the digital output of the accelerometer, a PAK VIIa chip was purchased. The PAK takes a PWM signal as an input and can count how long the signal is high or low. It then outputs a 16 bit value of how long the signal was high or low. Each bit represents 5 µs. This would have been a nice solution for us to measure the signals from the accelerometer. After multiple tests and communication with the PAK manufacturer, the chip was found to be defective. This forced us to use the analog method to retrieve values from the accelerometer.

3.3.3 Formulas

The correct formulas need to be used in order to go to the correct distances. In the end, these were the equations chosen to be used:

$v = v_0 + at$
$x = x_0 + v_0 t + \frac{1}{2} a t^2$

The third term (1/2 at^2) was not used since the difference in time was low, which made the result really low in comparison to the vt term.

Careful consideration went into coding these formulas due to the fact that the OOPic does not have floating point capability and to prevent overflow in any variable.

4 Compass

This section provides information about the compass which was chosen for the design of the Dead Reckoning Robot. The subsections below contain information about the device itself; hardware and software details of the OOPic interface and issues that arose during integration of the sensor.

4.1 Device Information

The compass chosen for use in this design is the Devantech CMPS03. It has a resolution of 0.1 degrees and an accuracy of 3 – 4 degrees. The heading of the robot can be obtained in one of two ways. A PWM signal can be read from pin 4 or the I2C interface can be used with pins 2 and 3.

- CMPS03 Product Page
    - http://www.robot-electronics.co.uk/htm/cmps3doc.shtml

4.2 OOPic Interface

This section describes the details of the interface to the OOPic microcontroller.

4.2.1 Physical Interface

This section provides the details of the physical electronic interface for this sensor.

4.2.1.1 Physical Location

The position of the compass was determined mainly due to the positions of the other sensors. The accelerometer and the gyroscope need to be as close to the center of the robot as possible. The compass was then placed in the middle of the robot, but closer to the front edge of the sensor PCB.

4.2.1.2 Pinout

The following table describes how the compass was connected to our 40 pin OOPic connector.

Note that the OOPic **PIN** is not the same number as an OOPic I/O line.

Compass Pin	Description	OOPic Pin
1	+5 VCC	21
2	SCL (I2C Interface)	3
3	SDA (I2C Interface)	1
9	GND	23

4.2.2 Software Interface

The use of the I2C interface was a huge advantage in coding for the compass. The bearing read by the compass is stored within registers and that value can be directly read thru the I2C interface. Registers 2 and 3 contain the 16 bit bearing and is sent to the OOPic high byte first.

The OOPic has an I2C object which is used to identify the compass. The following is sample code on how to get a bearing reading from the compass.

```
Dim compass As New oI2C       ' Create the compass objects
Dim heading As new oWord      ' Somewhere to store the reading



Sub Main()
  Compass.Node = 96           ' Decimal of Hex address 0xC0 shifted right by 1

  Compass.Mode = cv10bit      ' I2C mode is 10-Bit Addressing.

  Compass.NoInc = 1           ' Don't increment
  Compass.Location = 2 ' Address of single byte bearing

  Compass.Width = cv16bit     ' Compass Data is 2-byte wide.

  Compass.Location = 2 ' Address of word bearing

  Bearing = Compass.Value     'Get bearing from register

End Sub
```

4.3 Issues

This section describes problems the design team encountered.

4.3.1 Reliability Indoors

When testing the compass in the lab, an interesting anomaly was found. With the aid of a map compass, the lab floor was found to have many different directions giving a north reading. The magnetic field was never consistent anywhere in the room. We attributed this to the many computers giving some magnetic rays and to wires in the floor and ceilings. When the robot was tested outside, the reliability increased dramatically.

4.3.2 Powering the Compass

The compass needs a nominal 15 mA to operate. At times though, 400 mA current is needed. The OOPic power lines only provide 75 mA. Since the spike in current occurs so infrequently, this issue did not command much time. As of yet, no problems have yet to come to our attention, yet it is a slight concern.

5 Encoders

This section provides information about the encoder which was chosen for the design of the Dead Reckoning Robot. The subsections below contain information about the device itself; hardware and software details of the OOPic interface and issues that arose during integration of the sensor.

5.1 Device Information

The encoders chosen the e3 US Digital encoders. The link to the website is:

http://www.usdigital.com/products/e3/

These encoders have twice the resolution of the e2 encoders and were the perfect size for the MARK III robot. The link gives shows mechanical drawings and dimensions, further product description, and output explanation.

5.2 OOPic Interface

This section describes the details of the interface to the OOPic microcontroller.

5.2.1 Physical Interface

This section provides the details of the physical electronic interface for this sensor.

5.2.1.1 Physical Location

The encoders need to be placed on the wheels in such a manner that the wheels can move, but whole encoder doesn't. The only part of the encoder that rotates with the wheel is the thin plastic film found inside of the encoder.

5.2.1.2 Pinout

The following table describes how the encoders were connected to our 40 pin OOPic connector.

Note that the OOPic **PIN** is given not the OOPic I/O line.

Encoder Left Pin	Description	OOPic Pin
1	A	
2	B	
3	Index	
4	VCC (Power Supply)	

5	GND	
Encoder Right Pin		
1	A	
2	B	
3	Index	
4	VCC (Power Supply)	
5	GND	

The 2.5 V Precision Reference was not included in the connections to the OOPic because there were not enough 10 bit A/D inputs available. This could be connected to one of the 3 remaining A/D converters available on the OOPic.

5.2.2 Software Interface

The encoder values obtained are digital pulses that are converted to cm traveled by the encoders. Using these values, we can navigate the robot to a desired location or find how far a robot has traveled. The following code shows how to instantiate the converter object that makes it easy to read the output of the encoders. The oQencode object is part of the OOPic library. More information on this object can be found on the OOPic website.

```
 Dim encodeL As New oQencode
Dim encodeR As New oQencode

 Sub Main()
        encodeR.IOLine1 = 25
        encodeR.IOLine2 = 24
        encodeR.signed = 1
        encodeR.Operate = cvTrue

        encodeL.IOLine1 = 15
        encodeL.IOLine2 = 14
        encodeL.signed = 1
        encodeL.Operate = cvTrue
 End Sub
```

5.3 Issues

The only issue that the encoder had was mounting them on the wheels. It took a little planning and a little more effort than the rest of the sensors.

6 Gyroscope

This section provides information about the gyroscope which was chosen for the design of the Dead Reckoning Robot. The subsections below contain information about the device itself; hardware and software details of the OOPic interface and issues that arose during integration of the sensor.

6.1 Device Information

The gyroscope chosen for the design was the ADXRS300. This is a MEMS type gyroscope capable of measuring values of angular velocity in the range ± 300 °/s.

The design uses an evaluation board version of the gyroscope. The part number for this evaluation board is ADXRS300EB. This evaluation board includes several of the capacitors used to bias the gyroscope. Additionally the evaluation board comes in a standard 20 pin DIP package eliminating the time required for physically mounting the sensor on our PCB.

Analog Devices also makes a version of the gyroscope which measures ± 150 °/s. Using this devices would allow the angular velocity to be measured to a greater accuracy. The maximum angular velocity of the robot would have to limited though with this sensor.

The reason for this limitation is that when the wheels are turning in opposite directions the robot can easily exceed ± 150 °/s. Te described design choose to go with the lower resolution of the gyroscope with the larger measurement range.

The specifications for the gyroscope itself can be obtained from Analog Devices website. Links to Analog Devices' homepage, the product page for the gyroscope and the product page for the evaluation board are as follows:

- Analog Devices' Homepage
  - http://www.analog.com/
- ADXRS300 Gyroscope Product Page
  - http://www.analog.com/Analog_Root/productPage/productHome/0%2C2121%2CADXRS300%2C00.html
- ADXRS300EB Gyroscope Evaluation Board Product Page
  - http://www.analog.com/Analog_Root/productPage/productHome/0,2121,generic%253DADXRS300%2526level4%253D%25252D1%2526level1%253D292%2526level2%253D%25252D1%2526level3%253D%25252D1%2526metaId%253D2656%2526resourceWebLawID%253D356,00.html

6.2 OOPic Interface

This section describes the details of the interface to the OOPic microcontroller.

6.2.1 Physical Interface

This section provides the details of the physical electronic interface for this sensor.

6.2.1.1 Physical Location

The ADXRS300EB is a 20 pin DIP (dual inline pin) package. The board was placed near the center of rotation of the robot on the sensor PCB.

It is important that the gyroscope be placed as closely as possible to the center of rotation. The further the gyroscope is placed from the center of rotation the more inaccurate the readings from the device will be.

Because the accelerometer also needs to be on the axis of rotation it may be necessary to choose one sensor, either the gyroscope or the accelerometer, to be exactly on the axis of rotation forcing the other to be slightly offset.

6.2.1.2 Soldering

During initial testing the evaluation board was not soldered to the PCB. This caused the wires which were wrapped to the pins to come loose repeatedly. The solution was to solder on the four outer pins to prevent the gyroscope to vibrating during operation of the robot.

Only the outer four pins were soldered so the device could easily be relocated if necessary.

6.2.1.3 Pinout

The following table describes how the gyroscope evaluation board was connected to our 40 pin OOPic connector.

Note that the OOPic **PIN** is give not the OOPic I/O line.

Gyroscope Pin	Description	OOPic Pin
1	AVCC (+ Analog supply)	22
2	RATEOUT (Rate Signal Output)	7
7	2.5 V (2.5 V Precision Reference)	NC
8	AGND (Analog Supply Return)	24
9	TEMP (Temperature Voltage Output)	11
10	ST2 (Self-Test for Sensor 2)	10
11	ST1 (Self-Test for Sensor 1)	12
12	PGND (Charge Pump Supply Return)	23

13	PDD (+ Charge Pump Supply)	21

The 2.5 V Precision Reference was not included in the connections to the OOPic because there were not enough 10 bit A/D inputs available. This could be connected to one of the 3 remaining A/D converters available on the OOPic.

6.2.2 Software Interface

The gyroscope provides a simple analog output which represents the angular rate. Basically a value of 2.5 V indicates a rate of 0 º/s. An change in voltage on the RATEOUT output indicates a linearly related change in angular velocity. The value of the voltage relative to the 2.5 V bias value indicates the direction of the angular velocity.

Specific details on interpreting the values of the gyroscope can be found in the specification for the ADXLS300 available from Analog Devices' website.

The following code shows how to instantiate the A/D converter object necessary to read the output of the gyroscope. The oA2D10 object is part of the OOPic library. More information on this object can be found on the OOPic website.

```
Dim rate As New oA2D10
Dim temp As New oA2D10
Dim rateAnalog as New oWord


Sub Main()
  rate.IOLine = 1
  rate.Operate = cvTrue
  temp.IOLine = 2
  temp.Operate = cvTrue
End Sub
```

6.3 Issues

This section describes problems the design team encountered.

6.3.1 Powering the Gyroscope

The specifications for the gyroscope indicate that the analog supply is sensitive to noise, especially to that generated by digital circuits. The specification recommends that this power input be isolated from other devices.

Because of the limited number of power connections available in this design the analog power inputs for the gyroscope where in parallel to several other devices. This did not cause a problem during testing but is something that the user should be aware of.

User Manual:
Dead Reckoning Robot

Senior Design
Spring 2003
Texas A&M University

Lee Fithian
Ajay Joseph
Steve Parkinson
Saba Rizvi

1 Introduction

This document is intended for those who wish to construct a replica of the Dead Reckoning robot. This robot uses a Mark III robot chassis with the OOPic microcontroller and integrates four different sensors which are useful for the development of dead reckoning algorithms.

The four sensors integrated into this robot are:

- MEMS Accelerometer
- Compass
- Shaft Encoders (1 per wheel)
- MEMS Gyroscope

This document does not discuss issues concerning the design of the Dead Reckoning robot. For detailed information on the implementation used refer to the Implementation Notes document.

There are several aspects of this implementation which could be greatly improved upon. The reader is encouraged to study the specifications of each of the sensor and improve upon the design contained in this document.

1.1 Audience

This document describes the basics of constructing the Dead Reckoning robot and interfacing the included sensors from the OOPic microcontroller. Anyone who can use simple tools such as a saw, wire wrapper and drill should be able of completing the construction of this robot.

All the code contained in this document is written in Visual Basic for the OOPic. This is almost identical to Microsoft Visual Basic. This document assumes that the reader has an understanding of the basics of programming and a basic understanding of the syntax of Visual Basic.

1.2 Section Descriptions

The sections of this document follow a chronological order of assembling the Mark II robot, physically integrating each of the sensors and then interfacing the sensors from software.

The first section provides a list of the parts you will need to order for construction. When possible online retailers which can supply these parts and an approximate cost at the time of writing is included.

The section describes the tools that will be required to complete the assembly. This list does not include those tools which the Mark III robot assembly instructions.?????

The next sections describe the physical integration of each of the sensors with the Mark III kit. The reader is encouraged to consider the design presented and include improvements of their own design.

Following the physical integration sections, several sections providing code examples used to interface the sensors are included.

Because the majority of the time in the development of this project was spent physically integrating the sensors to the device the algorithms used are basic and certainly not optimal. The reader is again encouraged to evaluate their intentions for the robot and improve upon the software presented.

2 Required Parts

Mark III Complete Kit (OOPic Version)	$98.00
OOPic II+ Upgrade Kit	$30.00
Serial Extension Cable	$4.00
Accelerometer Kit	$23.00
Devantech CMPS03 Digital Compass	$38.00
http://www.junun.org/MarkIII/Store.jsp	

E3 Optical Kit Encoder (2) E3-64-197-HM-PKG3	$66.00 Each
http://www.usdigital.com/	

Angular Rate Sensor ADXRS300 (Gyroscope)	$50.00
http://www.analog.com/	

Accelerometer Socket LCC81.2701	$20.50
http://www.el-mech.com/	

Low Cost Prototype Circboard V2009-ND	$10.47
40 Pin Connection Header WW CHW40G-ND	$5.43
http://www.digikey.com	

40-Cond. Ribbon Cable	$1.17
Female Connector for Ribbon Cable (2)	$2.56 Each
6 Pin Locking Connector (2)	$1.59 Each
6 Pin Locking Header	$0.99
Heat Shrinkable Tubing	$1.73
Mid-State Electronic Supply, Inc. – Bryan, Tx	

4-40 x 36” All Thread Rod	$2.44
4-40 Machine Screw Sleeve (2)	$0.15 Each
Ace Bolt and Screw Co. – Bryan, Tx	

18x6 Sheet Metal	$3.93
4-40 x 1” Combo Nut and Bolts	$0.83

6-32 Hex Mach Nuts	$0.83
Lowe’s – Bryan, Tx	

9V DC 300mA Adapter	$10.99
Radio Shack – College Station, TX	

4 AA Rechargeable Batteries	
9V Ni-MH Rechargeable Battery	
Wire-Wrapping Wire	
Provided in Lab	

3 Required Tools

Soldering Iron	Wire Cutter
Wire Wrapper	Ruler
Digital Multimeter	Level
Needle Nose Pliers	AA Recharging Base
Screwdrivers	9V Recharging Base
Saw	Dremel
Drill	

4 Physical Integration of Sensors

This section gives instructions on physically connecting each of the sensors to the OOPic microcontroller. The implementation described mounts the sensors on a pad per hole PCB mounted above the OOPic microcontroller.

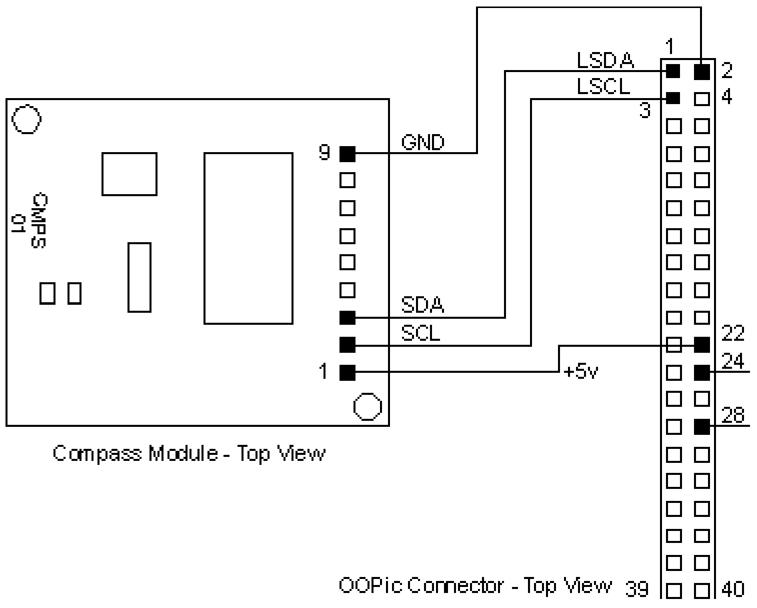
4.1 Accelerometer

The accelerometer was purchased in a kit which provided the team with the actual accelerometer as an LCC and various resistors and capacitors in order to tune the outputs and set the bandwidth of the signal. Since it is an LCC, a socket was required to mount it to the PCB. A suitable socket was eventually found, but it did present a problem. The pins on the bottom of the socket do not line up to a standard PCB layout. To solve this problem, the PCB was altered by drilling slots to accommodate for the non-standard pins. Once this was done, the socket rested level on the PCB and wires could be either soldered or wrapped onto the pins.

From those accelerometer socket pins, the wires were then connected to another socket which contains a circuit of the aforementioned resistors and capacitors. This was done so that the accelerometer can be tuned differently for future project with ease. For detailed circuit diagrams, consult the physical integration of the accelerometer in the Implementation Notes.

The analog outputs of X filter (pin 7) and Y filter (pin 6) are connected to the OOPic and the distance traveled can be measured off these signals.

4.2 Compass



The compass is oriented so that the north side (left side in the picture) is facing the direction the robot will be moving forward in. Only 2 output lines are required and are connected to the pins 1 and 3 of the OOPic.

4.3 Encoder

The encoders must be mounted on the wheels in such a way that the wheels will rotate without rotating the entire encoder. The only thing that is rotated is the thin film inside of the encoder. Each encoder has five outputs: a, b, index, voltage, and ground. These outputs can be found at the top of the encoder.

4.4 Gyroscope

The gyroscope we used is an Analog Devices ±300°/s Single Chip Yaw Rate Gyro with Signal Conditioning, ADXRS300. We purchased an evaluation board, ±300°/s Single Chip Rate Gyro Evaluation Board, ADXRS300EB, to provide the necessary circuitry to tune the device.

The evaluation board is a standard .3” by .1” dual-in-line mount. It has 20 leads with a PCB mounted on top. Capacitors have been installed for basic signal filtering and bandwidth setting (default bandwidth is 40 Hz). The ADXRS300EB is not reverse polarity protected, and may by damaged by applying inappropriate voltages to any pin.

Below are the pin connections between the ADXRS300EB and the OOPic, Table 1:

Gyroscope Pin	Description	OOPic Pin
1	AVCC (+ Analog supply)	
2	RATEOUT (Rate Signal Output)	
7	2.5 V (2.5 V Precision Reference)	
8	AGND (Analog Supply Return)	
9	TEMP (Temperature Voltage Output)	
10	ST2 (Self-Test for Sensor 2)	
11	ST1 (Self-Test for Sensor 1)	
12	PGND (Charge Pump Supply Return)	
13	PDD (+ Charge Pump Supply)	

A diagram of the circuit is in below (Figure 1):

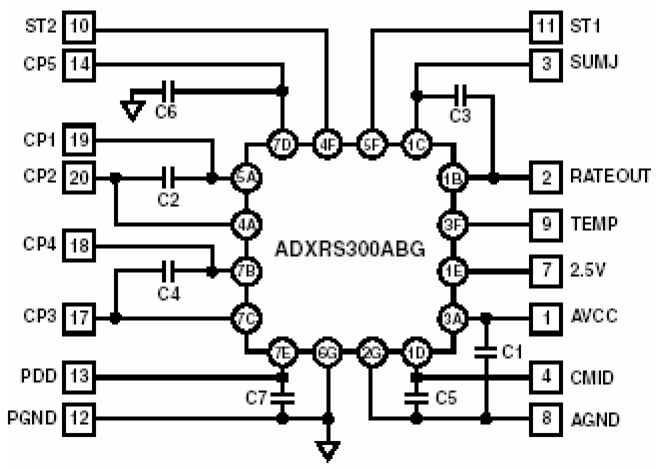


Figure 1. ADXRS300EB Schematic

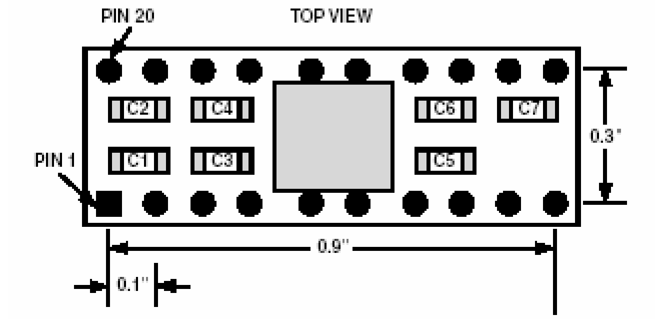


Figure 1

5 Software Integration of Sensors

This section discusses the software used to read raw data from each of the sensors using the OOPic microcontroller.

5.1 Accelerometer

This design uses the analog output from the accelerometer. Using the accelerometers in analog is described thoroughly in the specifications for the device. The OOPic object used is the oA2D10. The oA2D can also be used.

The OOPic microcontroller website has information concerning the use of these objects.

5.2 Compass

The software interface for the compass uses the I2C interface of the OOPic microcontroller. Reading the value of the compass is thoroughly documented by Devantech.

5.3 Encoder

The encoder values obtained are digital pulses that are converted to cm traveled by the encoders. Using these values, we can navigate the robot to a desired location or find how far a robot has traveled.

In order to get the bearing from the compass, an I2C object is used to retrieve the value from a register located on the compass. The value is represented in 16 bits, which allows for a resolution of 0.1 degrees. In other words, a bearing of 0 – 359.9 can be read. Here is some example code. This program the robot is given an x direction and y direction and is asked that is simply reads the register to travel these distances using the encoders. The robot will first travel the x direction, then turn 90 degrees obtain the direction of what y needs to go (positive or negative) and finally travels the y direction given.

```
Dim encodeL As New oQencode       'Declare all variables to be used in program
Dim encodeR As New oQencode
Dim servoL as New oServoSP1
Dim servoR as New oServoSP1

Dim fwdPls as new oWord
Dim encTmp as new oWord

dim holdL as new oWord
dim holdR as new oWord

dim x as new oWord
dim y as new oWord

dim stage as new oByte

Sub SetXY()
```

```
  x = 60                'set x and y to how many centimeters you want it to move
  y = -60
End Sub

Sub Main()
  Setup
  OOPic.delay = 400
  fwdPls = (x * 61) / 10         'The number that will be multiplied to each
encoder value to convert pulses to
                                           'cm
  if (x > 0) then
    servoR = 30                     'Set servo speed to start moving
    servoL = 30
    Do                                                   'Do loop until the
destination x is near
      encTmp = encodeR - encodeL
      if (encTmp > 20) then                    'This if else block is used to keep
the robot on a straight path
        servoL = 120
        servoR = 60
      elseif (encTmp < -20) then
        servoR = 120
        servoL = 60
      else
        servoR = 120
        servoL = 60
      end if

      encTmp = (encodeR + encodeL) / 2     'Find distance traveled so far
      OOPic.delay = 10
    Loop while fwdPls > (encTmp + 30)

    servoL = 30
    servoR = 30
    Do            'This do loop starts when destination x is near and slows
                  'down the servos until destination x is reached
      encTmp = (encodeR + encodeL) / 2
    loop while fwdPls > encTmp
  end if
  stage = 1
  servoR = 0
  servoL = 0
  OOPic.Delay = 50
  holdL = encodeL.value
  holdR = encodeR.value
  encodeR.value = 0
  encodeL.value = 0
  stage = 2 'turning part
  if (y > 0) then 'This part handles the y distance. It first turns the robot 90
                  'degrees in the desired y direction and travels the y distance
                  'in a similar fashion as the x
    stage = 3
    servoR = 30
    servoL = -30
    Do
      OOPic.Delay = 1
    loop while encodeR < 28
  end if
  if (y < 0) then
    stage = 4
    servoL = 30
    servoR = -30
    stage = 5
    Do
      OOPic.Delay = 1
    Loop while encodeL < 36
  end if
  stage = 6
  servoR = 0
  servoL = 0
  OOPic.Delay = 50
  encodeR.value = 0
  encodeL.value = 0
  OOPic.Delay = 50
  fwdPls = abs((y * 61) / 10)
  if (y <> 0) then
    servoR = 120
    servoL = 120
    Do
      encTmp = encodeR - encodeL
      if (encTmp > 20) then
        servoL = 120
        servoR = 60
      elseif (encTmp < -20) then
       servoR = 120
       servoL = 60
      else
        servoR = 120
        servoL = 60
      end if
      encTmp = (encodeR + encodeL) / 2
      OOPic.delay = 10
    Loop while fwdPls > (encTmp + 30)
    servoR = 30
    servoL = 30
    Do
     encTmp = (encodeR + encodeL) / 2
    loop while fwdPls > encTmp
  end if
  servoR = 0
  servoL = 0
  OOPic.operate = cvOff
End Sub

Sub Setup()      'This subroutine is called in main. It sets the servos
                 'and encoders up to be used in the program
  x.signed = cvTrue
  y.signed = cvTrue
  SetXY
End sub
```

To integrate the device: Insert the device into the PCB where desired, and connect to the appropriate pins from the ADXRS300EB to the OOPic input bank. The leads on the ADXRS300EB are not long enough to go through the PCB and be suitable for wire wrapping. Soldering the connecting wires is necessary to provide stable connections. Make sure to not overheat the ADXRS300EB while soldering.

Tip: Put the gyroscope in a position near to the moment about which the robot will turn. This will provide more accurate measurements.

5.4 Gyroscope

The gyroscope works on an output voltage that is proportional to the angular velocity.

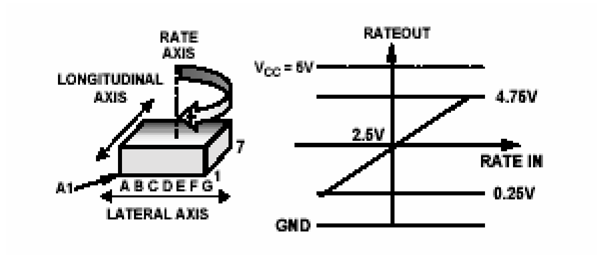


Figure 2. RATEOUT Signal Increases with Clockwise Rotation

```
Dim rawRate As New oA2D10
Dim rateOld as new oWord
Dim rateNew as new oWord
Dim cntOld as new oWord
Dim cntNew as new oWord
Dim zeroRate as new oWord
Dim angDisp as new oWord

Dim cnto As New oCounter
Dim count As New oWord
Dim clk as new oClock
```

The voltage can be scaled, and works with the equation:

$$\text{Angular Velocity} = (\text{Rateout} - 2.5)/2.25 *300$$

```
Dim servoL as New oServoSP1
Dim servoR as New oServoSP1

const destAngle = 900



Sub Main()
  StartVCs
  Setup
  cntOld = count
  rateOld = ((125 * ((rawRate.Value + zeroRate) / 3)) / 64) - 333
  if (destAngle > 0) then
    servoL = 30
    servoR = -30
  elseif (destAngle < 0) then
    servoL = -30
    servoR = 30
  end if
  do
    cntNew = count
    rateNew = ((125 * ((rawRate.Value + zeroRate) / 3)) / 64) - 333
    if(cntNew > cntOld) then
      cntOld = cntNew - cntOld
    else
      cntOld = (65535 - cntOld) + cntNew
    end if
    angDisp = angDisp + ((((rateNew * cntOld * 10) / 283)*2))
    cntOld = cntNew
    rateOld = rateNew
  loop while ( angDisp < (destAngle))
  servoL = 0
  servoR = 0
End Sub

Sub Setup()
  servoL = 0
  servoL.IOLine    = 10
  servoL.InvertOut = cvFalse
  servoL.Operate   = cvTrue

  servoR = 0
  servoR.IOLine    = 9
  servoR.InvertOut = cvTrue
  servoR.Operate   = cvTrue

  angDisp.signed = cvTrue
  angDisp = 0

  rateOld.signed = cvTrue

  rateNew.signed = cvTrue

  rawRate.IOLine = 1
  rawRate.Operate = cvTrue

  OOPic.Delay = 100

  zeroRate.signed = cvTrue
  zeroRate = rawRate.value
  zeroRate = 512 - zeroRate
End Sub

Sub StartVCs()
  clk.Rate = 255
  clk.Operate = cvTrue
  cnto.Output.Link(count.value)
  cnto.Clockin1.Link(clk.Result)
  cnto.Operate = 1
End Sub
```

6 Getting Help

This section discusses some resources for getting help with the devices used in this project.

6.1 *Mark III and OOPic Yahoo Groups*

These 2 resources were used many times. Searching through the messages stored in these groups proved to be very valuable since many of the issues we had encountered have been seen and in many cases solved in previous attempts.

http://groups.yahoo.com/group/MiniSumoMarkIII/
http://groups.yahoo.com/group/oopic/

6.2 *Analog Devices*

In addition to getting datasheets and product specifications from this site, the technical support for the company assisted the team in finding an appropriate socket for the ADXL202E LCC.

We are able to use software objects provided by Savage Innovations to read the ADXRS300. We use an A2D10 object that converts a voltage to a 10-bit digital value. The value the ADXRS300 provides corresponds the amount of angular velocity the gyroscope is experiencing. The angular velocity can be determined by finding the location of the voltage on the scale of possible voltages the gyroscope operates over. At the default voltage, the gyroscope experiences 0 °/s and outputs a voltage of about 2.5 V. As the gyroscope turns, the voltage will vary, indicating that the gyroscope is turning. As the gyroscope turns clockwise, the voltage will increase and angular velocity increases. The gyroscope will have a value of 4.75 V when the maximum value of 300 °/s is achieved. The same is true for counter-clockwise rotation, except the voltage approaches .25 V with a minimum value of -300 °/s.
http://www.analog.com/

6.3 *Devantech*

This manufacturer provided detail information on how to interface the compass to the OOPic processor. Example code for the compass is available to download also.

The voltage is converted to the angular velocity as it is read. It then converted into the amount of angular displacement experienced by the gyroscope by multiplying by the time it experienced the angular velocity. The time is found using the virtual circuit explained in the Software Integration Section of the Accelerometer.

Below is the conversion of the voltage to the angular velocity and then into the angle turned through during the last time period (VB sample code):

```
        rateNew = ((125 * ((rawRate.Value + zeroRate) / 3)) / 64) - 333
```

http://www.robot-electronics.co.uk/htm/cmps3doc.shtml
http://www.robot-electronics.co.uk/htm/cmpsoopic.shtml

6.4 Al Williams

When we first tested the PAK VIIa, it did not work as was expected. Al Williams was then contacted to help in troubleshooting all possible scenarios. Quick responses were received from Mr. Al Williams himself. He gave the team various things to test, but in the end the chip was found to be defective.

```
      angDisp = angDisp + ((((rateNew * cntOld * 10) / 283)*2))
```

Where zeroRate is the standing voltage on the ADXRS300, rawRate.value is the current rate on the ADXRS300, and cntOld is the amount of time that has elapsed between the rate readings.
http://www.al-williams.com/

6.5 Savage Innovations (OOPic Microcontroller)

In order to fully understand all the capabilities of the OOPic microcontroller, this site was consulted many times. Important details such as pin descriptions, object definitions and programming syntax were all clearly explained on this website.

The conversion of the rawRate is the equation governing the output voltages of the gyroscope. More information can be found in the data sheet. The division by: 283 is done to convert the time into seconds; because the clock runs at 283 Hz, and yields 3.5 ms units for the cntOld value.

$$\theta = (\text{EncoderLeft} - \text{EncoderRight}) / \text{Wheel Base}$$

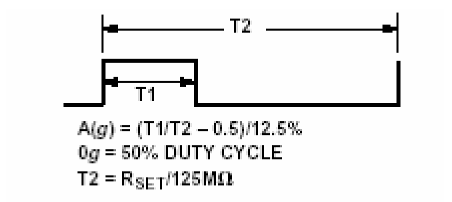
With the above conversions, the angular displacement can be accumulated and the angle turned through can be monitored. Below is a sample program to turn through 90°.

Then by using the distance traveled and the angle traveled, the x, and y distances can be found.

7 Appendix A: Sensor Descriptions

7.1 *Accelerometer*

The accelerometer gives two important values T1 and T2 that can be obtained from the output signal. T2 is a constant and will always remain the same. When T1 is half of T2 it will be at 1 g acceleration.



Using the equation show below the picture, the acceleration can be easily found.

7.2 *Compass*

The compass is very easy to use. It has a digital output that gives us the horizontal component of the earth's magnetic field. This value can also be used to turn angles or a sensor the robot can use to find it's location relative to an angle.

7.3 *Encoders*

Each encoder produces a counter value which is then multiplied by a constant equal to the number of meters per pulse the encoder moves. The calculated values represent the distance each encoder has traveled and the total distance of the robot can be found by:

$$\text{Distance} = (\text{EncoderLeft} + \text{EncoderRight})/2$$

Throughout the course of this entire project, many problems arose. From those problems, questions had to be answered, and listed below are the sources to the answers to those questions. Thanks go out to all those who have helped in the completion of this project.

7.4 *Gyroscope*

The gyroscope provides an analog voltage output proportional to the angular velocity. Details on conversion from voltage to angular velocity can be found in the specifications for the device.

Dead Reckoning Group
Course Debriefing

Lee Fithian
Ajay Joseph
Steven Parkinson
Saba Rizvi

The goal of this debrief is to document the collective thoughts from the individuals about the workings of the group and a discussion of future considerations.

Management Style:

Style:
The management style we used was one of task identification, and then task distribution. The leadership role was not clearly defined.

The task identification was open to all who had thoughts on what needed to be done. We attempted to come up with tasks that were clear and well defined. They were to be completed for the next meeting. If they could not be completed, they had to have been attempted thoroughly to allow for discussion of future solution.

To distribute tasks, we used volunteering coupled with assigning tasks to able individuals. Tasks were first offered up to anyone, allowing interested persons to work on that subjects they wanted to do. If no-one volunteered, then we looked for the person most able to complete the task. In the beginning, each person stuck to one area of the project and worked on this only. However, everyone started switching tasks to different areas later on so that everyone could learn what was going on as a whole. Many times, two or more members would work together to accomplish a certain task to gain more perspective into getting the task achieved correctly. It provided more validation than if one person had done it alone.

We had no explicit leader. That position was clearly kept open so that anyone willing to assume it could. It seems that since we were all on equal footing experience and education wise, it would have undermined our faith in each other to put one of ourselves in charge. This trust proved to be useful knowing that tasks would be accomplished by the individuals and that we did not have to constantly look over the other's shoulder but work on our individual tasks. No leader also allowed us to select tasks we would work instead of them being assigned.

Project Redo:
If we were to repeat this project we would probably have changed how often we met, the structure of out meetings, and task assignment.

Meeting more often would have allowed us to work together on parts of the project we were unsure about individually.

Changing meeting structure to be more focused on task division and completion would have allowed us to stay on schedule.

In the beginning of this project, we did not do a good job at task assignment and setting goals. We developed a very good system halfway throughout the semester, but if we had set this from the beginning, we would have been able to finish the project sooner.

Safety/Ethical Concerns:

We were careful to construct a robot that did not provide opportunities for a person to get hurt. The robot was built with all safety precautions in mind. Anyone using a tool that could potentially cause harm to himself/herself or to anyone in the area, was instructed to be cautious by any and all group members in the vicinity.

We also did our best to develop our own design with the help of our advisors. We have tried to give credit where it is due, documented in the user manual/implementation notes.

Product Testing:

We have tested our robot. We have tested each sensor and combinations of the sensors working together. Trial and error experiments were the cornerstone of this project. We have found that our combined navigation algorithms provide more accurate movement than the sensors functioning independently. However, we believe that if we had more time to develop better algorithms of converting data from each sensor, we would have been able to get better results.

A relevant situation that we have not test our product in would be an elaborate set of destination points. This make become a problem because of the voltage that is drawn by the OOPic. The voltage will drop until the robot runs too low to continue proper motivation.

Additional verification might include running standardized movement tests. These tests would allow the product to be ranked against other products similar in nature. One test includes the UMBmark. This is a test developed at the University of Michigan by Borenstein and Feng.

Some of the products may have been slightly exaggerated on how well they work. For example, the compass advertisement claims that the compass will measure within 3-4 degrees. However, there were many times that the compass gave us very terrible readings.

Dead Reckoning Group
Data Merging

Lee Fithian
Ajay Joseph
Steven Parkinson
Saba Rizvi

1. Goal

The goal of this portion of our project is to merge data into a single source for navigation. The reason for doing this is to incorporate multiple sensors so that a more accurate dead reckoning navigation system is produced.

2. Theory

Our method of merging data is composed of two core ideas. The first is a weighting system for each data input. The second is a way to combine the values in such a manner that the navigation only sees one input.

Weighting System

Our weighting system is based upon percent errors found for each sensor.

The percent errors were calculated based upon a basic testing algorithm described below. The testing would yield values from each sensor. We would then perform conversions on the sensor values. Each sensor had its own conversion.

Accelerometer conversion:

```
velocity = (((((avgRate - zeroRate) * 980) / 63) * time) / 283)
distTrav = distTrav + ((velocity * time * k) / 283)
```

Where velocity is that of the robot, avgRate was the average of all previous positive accelerations, zeroRate was the base value of the accelerometer, time was the amount of time elapsed between the measurements, and distTrav was the amount of distance the robot traversed.

Gyroscope conversion:

```
rate = ((125 * ((rawRate + zeroRate) / 3)) / 64) – 333
angDisp = angDisp + ((((rate * time * 10) / 283)*2))
```

Where rate was the angular velocity of the gyroscope, rawRate was the value read from the gyroscope, zeroRate was the base rate of the gyroscope, time was the amount of time elapsed between measurements, and angDisp was the angular displacement experienced by the gyroscope.

Once we had the converted values, we could determine the percent errors of each sensor, using the following equation:

$$\% \text{ Error} = (\text{Test} - \text{Actual}) / \text{Actual} * 100$$

From here, we used each sensors percent error to determine the other sensors weights.

Since we used pairs of sensors for the different types of movements (Translational and rotational), we exchanged the percent errors for the pairs and used the values as weights.

Combining Equation

In order to combine the values produced by the sensors used the following equation:

$$((\text{Sensor}_1 * \text{Weight}_1) + (\text{Sensor}_2 * \text{Weight}_2)) / \text{Target} < 1$$

Where each sensor is multiplied by its weight to determine its contribution to the total reading. Then, the total reading is divided by the target value. This division will approach one and pass it as the total reading passes the target value.

Using these two ideas, we were able to create a value for the navigation system to use in place of the individual sensors.

3. Experimentation

We developed algorithms that would move the robot to known distances/angles. We would measure each sensor value before and after the movement of the robot.

To setup these algorithms, we would use a local clock to determine when the robot should turn the servos off. We would then run trials to determine what value the local clock needed to reach in order for the desired distance to be traveled or rotation to occur.

Once we had our algorithms, we were able to monitor the values of the sensors through the OOPic Multi-Language Compiler.

4. Results

Trial Values

See the Appendix for results for the different experiments.

Weights

The experimentally determined weights we found are as follows:

Gyroscope:
- CW: 11

- CCW: 19

Encoders:

- Rotational
  CW: 89
  CCW: 81
- Translational: 88

Accelerometer:

- Translational: 12

5. Conclusion

The results show that the encoders are the dominant value in our data merging. We believe this to be acceptable because the encoders were easy to integrate and to monitor.

The lower weights on the gyroscope and accelerometer were most likely due to the interfaces we implemented. Each sensor was sensitive enough to provide a high level of accuracy. However, our interfaces were not standard interfaces with these devices. The gyroscope was sensitive to power fluctuations. The accelerometer would have been more accurate if the OOPic clock could have been faster. This would have helped because we could have more accurately measured the time that the pulse was high in the pulse-width modulated signal.

Another limiting factor for both gyroscope and accelerometer was the lack of floating point on the OOPic. This limited how we were able to convert the data.

Appendix

Translational Testing

Rotational Testing