Department of Electrical Engineering

University of Florida

INTELLIGENT MACHINES DESIGN LABORATORY

EEL 5934 (ROBOTICS)

Final Report on Gopher

Instructor: Dr. Keith L. Doty

Prepared by

MASOUD KAVEHZADEH

May 1, 1995

Table of Contents

ABSTRACT

In this project, a Motorola S68HC11EVBU board is used to design and build a relatively complex electronically sensualized autonomous robot called Gopher. When completed, this robot will be capable of starting from a physical location, called base, and service several other locations, called stations, by stopping at those stations and allowing for equipment or material to be loaded. Once it receives the signal to proceed it will continue on to the next station, and will repeat the process again. After the last station it will return to the base and wait for the new instructions.
Gopher was designed and assembled by me with the valuable help of the T/A'a and my classmates.

EXECUTIVE SUMMARY

This robot (Gopher) in the final stage will be able to accept A sequence of numbers corresponding to different stations that the robot needs to stop at. At these stations there may be signaling devices that by means of IR or Ultrasound transmitters provide information for location identification. It is also feasible to have markings on the floor to identify these locations to the robot.

On board Infrared or Ultrasound receivers will play part in the task of station recognition as well.

At the time of this report, far from its completed stage, Gopher's behavior is limited to straight line travel, and object avoidance.

Introduction

The first time I felt the need to have robots moving material around was in a computer warehouse in Miami.

In this warehouse in order for an invoice to be shipped, one of the seven or eight workers had to walk through this vast space look for and carry back each item on that invoice.

It took every worker most of its time just to walk back and forth to the location of each one of these items.

It was obvious that having small robots roaming around this warehouse and two or three workers loading and unloading parts on these robots, could greatly improve the speed of this operation.

Integrated system

Building Gopher involves building a platform, attaching two motors, three different types of sensors, and the memory enhanced Motorola 68HC11 board.

Once the assembly is complete the programming of the 68HC11 board will create the desired behavior for Gopher. All the stages of the construction of Gopher will be discussed individually.

Mobil Platform

Perforated aluminum was used as the material for Gopher's platform. A rectangular aluminum piece from the EE shop was obtained, cut , and sanded. In about 2 hours in the Low's hardware store the material to attach the wheels to the Futaba motors and

consequently to the platform were located. Two plastic round spacers (3/4" X 1/2") with two perpendicular half way cuts on the surface of each one of these spacers provided the interface between the wheels and the Futaba motor gears. The plastic parts were screwed and fit on the Futaba motors on the other sides of these spacers. Finally, using two longer screws the spacers were attached to the wheels It took about 9 hours to finish the platform with the motors attached.

The parts listed in table 1 were used to build the platform and assemble the motors, wheels, battery holders, and mount the EVBU board.

Parts Description	Size	QTY	PURPOSE
Perforated aluminum		1	platform
Wheels off the Radio Shack		2	driving wheels
Roller caster		1	front wheel
Nylon flat #4	6-32 X 1	4	
Nylon spacer	1/2 X .257 X 1/2	2	inside wheels #4 washers
Nylon finishing washer	.580 X .191	4	roller caster
Nylon wing nut	6-32	4	roller caster
Nylon spacer	1 x 3 1/8 x 3/8	2	wheel connection
Cone nuts	6-32	2	wheel connection
Philips flat head	10-24 x 1/2	4	roller caster
Stainless nylon insert log nut	10-24 18-8	4	roller caster
Slotted round head notch Screw	#4-40 x 1 1/2	2	wheels
Stainless hex nut	#4-40	2	wheels
Flat washer	#4	2	wheel to motor attachment
Aluminum strips		4	motor support brackets

Table 1 - Parts used for the platform assembly.

Actuation

Two Futaba motors, model No. FP-S148, which came with pre-assembled gear mechanism are the only actuation mechanism on Gopher. These motors have to be modified to allow for continuos rotation as opposed to partial rotation that they were originally designed for. The modification is depicted in figure 1.

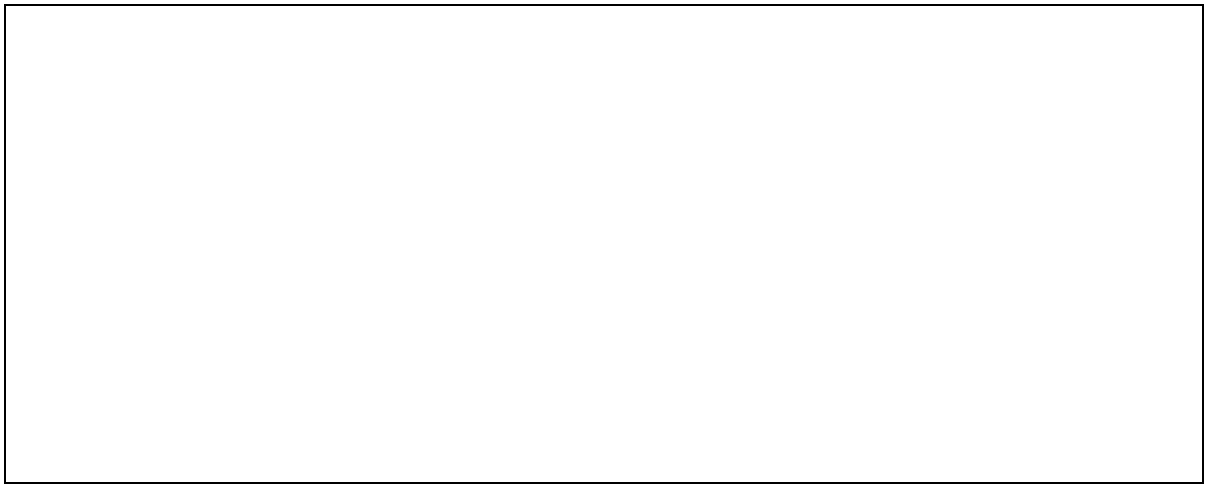

Fig. 1 - Modification of the Futaba PC board

Next the supporting electronic circuits to accommodate the IC program and the motor control circuitry were implemented. RAM chip, Latch, Hex Inverter, NAND gate and the motor driver are some of the chips installed on the EVBU board. All wire wrapping except the one for the Hex Inverter was done according to the schematic in Jones and Flynn book. The only deviation from the Jones and Flynn book was the use of 74HC04 Hex Inverter instead of 74HC4049. The reason for the switch was simply availability of the parts.

The first try in wiring the 32K RAM chip proved to be unsuccessful.

It turned out that the port C pins were wired wrong. With the help of Scott (our T/A) who immediately noticed that pin 10 is not connected the problem was solved in less than one hour.

Fig. 2 shows the schematic used to upgrade EVBU board with 32K extended RAM.

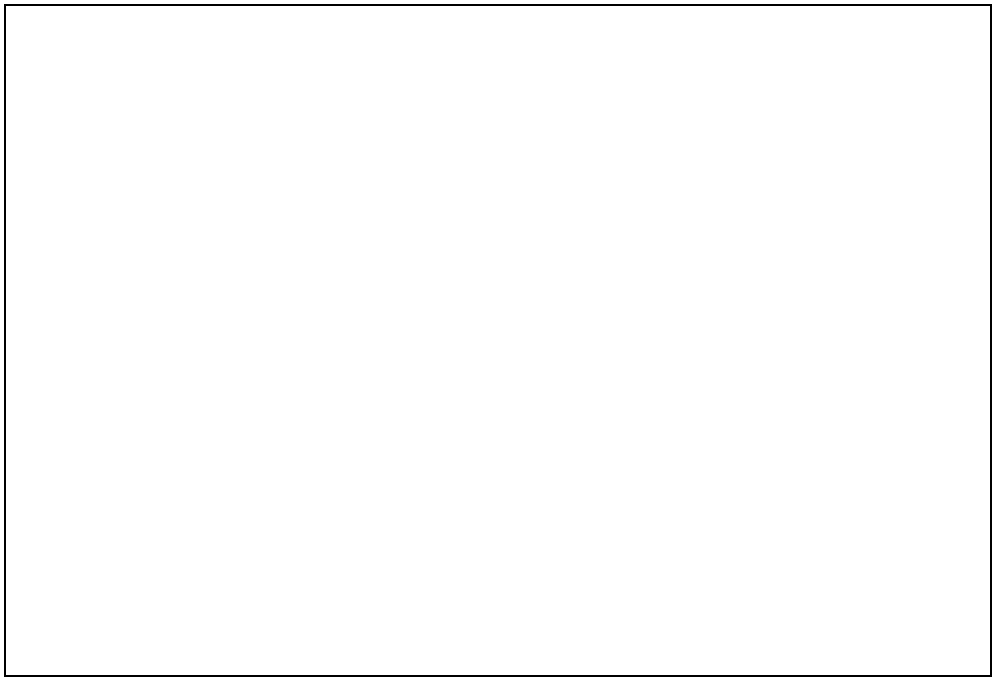

Fig. 2 - Static RAM (62256ALP) and Latch (74HCT573) wiring schematic

The wiring schematic for the L293NE motor driver is shown in figure 3.

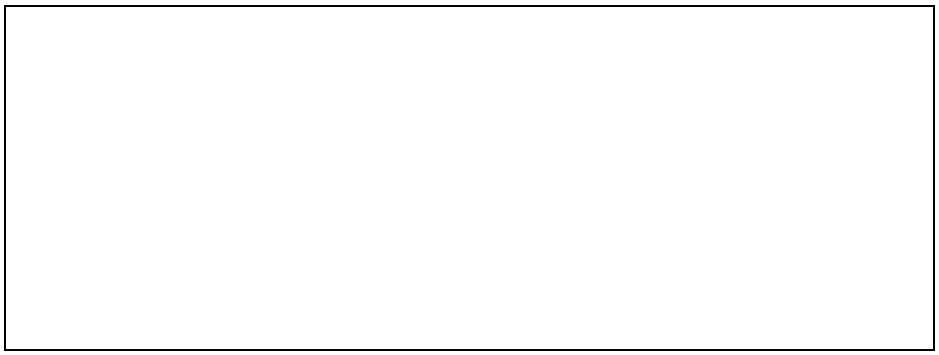

Fig. 3 - Wiring schematic for the L293NE motor driver

Electronic components used in the first phase of the project are listed in table 2.

Quantity	Description
1	62256ALP-10 32K Static RAM
1	74HCT573 Octal Latch
1	74HCT10 Triple 3-Input NAND
1	74HC04 Hex Inverter
1	L293NE Motor Driver
1	14-Pin wire-wrap socket
2	16-Pin wire-wrap socket
1	20-Pin wire-wrap socket
1	28-Pin wire-wrap socket
1	1K Resister
2	4-AA battery holder
1	470 μF electrolytic capacitor
7	0.1 μF bypass capacitor
1	Red LED
3	36-pin male header
1	36-pin female header
2	Siemens IR Emitter and Detector -

Table -2 Electronic components used for phase one

SENSORS

There are a total of three different sensors mounted on Gopher. These sensors are IR shaft encoders, IR Sharp sensors, and Ultrasound sensor.

Figure 4 shows the location of the sensors on Gopher.

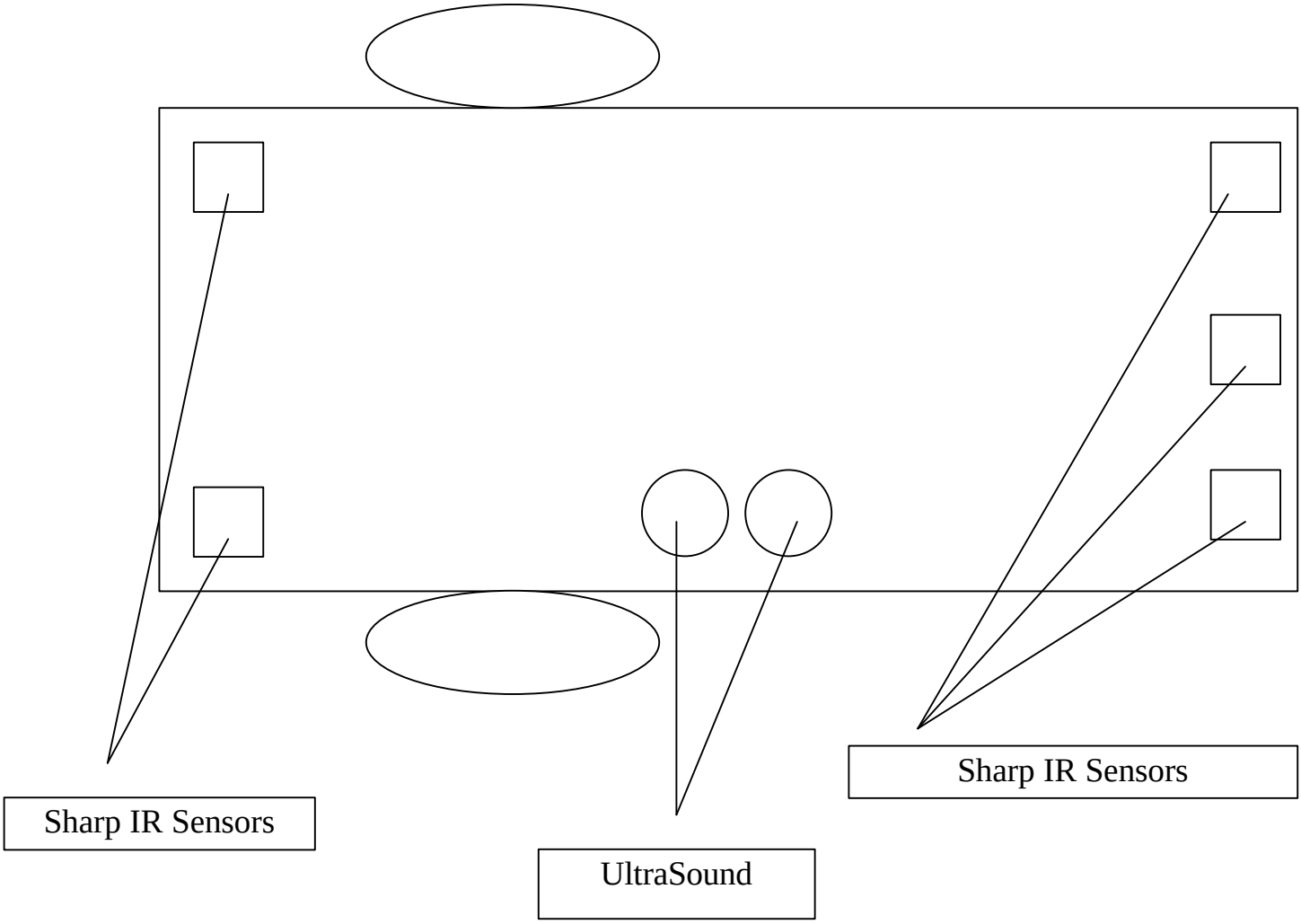

Fig 4 - Location of sensors on Gopher

1- Shaft Encoders

The shaft encoder IR's (infrared emitter and detector) were the first sensor mounted on Gopher. Following the schematic supplied with the shaft encoders specification sheet was

not a good idea. The schematic was wrong. Pin 2 (the middle one) is the ground pin and not the pin 3 (the furthest one) that the schematic suggests. It that seems every body else had followed the drawings done by our other T/A Bruce and they didn't go through the confusion that I did.

The shaft encoders were installed based on Dr. Doty's suggestion in the Futaba motor casings. In order to make room for these IR sets, the leads of the potentiometer on the Futaba PC board had to be cut. It is important for these pots to stay in place. Since there are four small plastic grips holding these pots in place; I don't believe that cutting the leads reduced the support for these pots.

The wiring schematic for the shaft encoders is shown in figure 5. Additional 3 wires were used for the shaft power and signal leads. The existing power and ground wires on the Futaba motors could not be used because of the changing polarity on these wires for direction control.

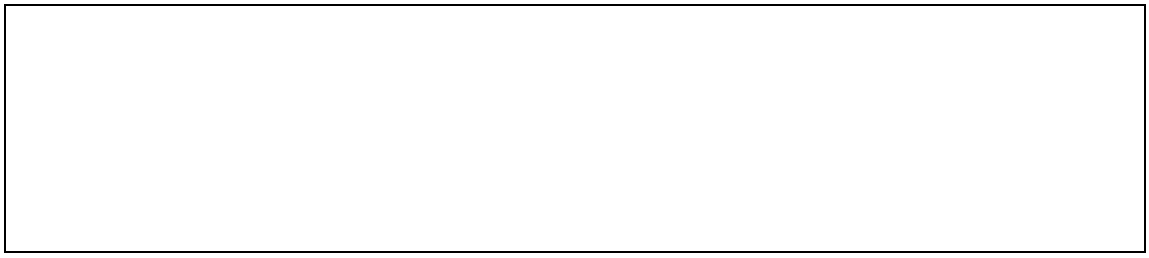

Fig. 5 - Wiring schematic for the infrared shaft encoders

2- IR SENSORS

The Sharp IR sensors obtained from the Robotics Lab are model GP1U5. These sensors can easily be modified to provide analog signal in addition to the digital signal that they are originally designed to produce.

Figure 6 shows this modification to Infrared Sharp sensors model GP1U5 series 8x.

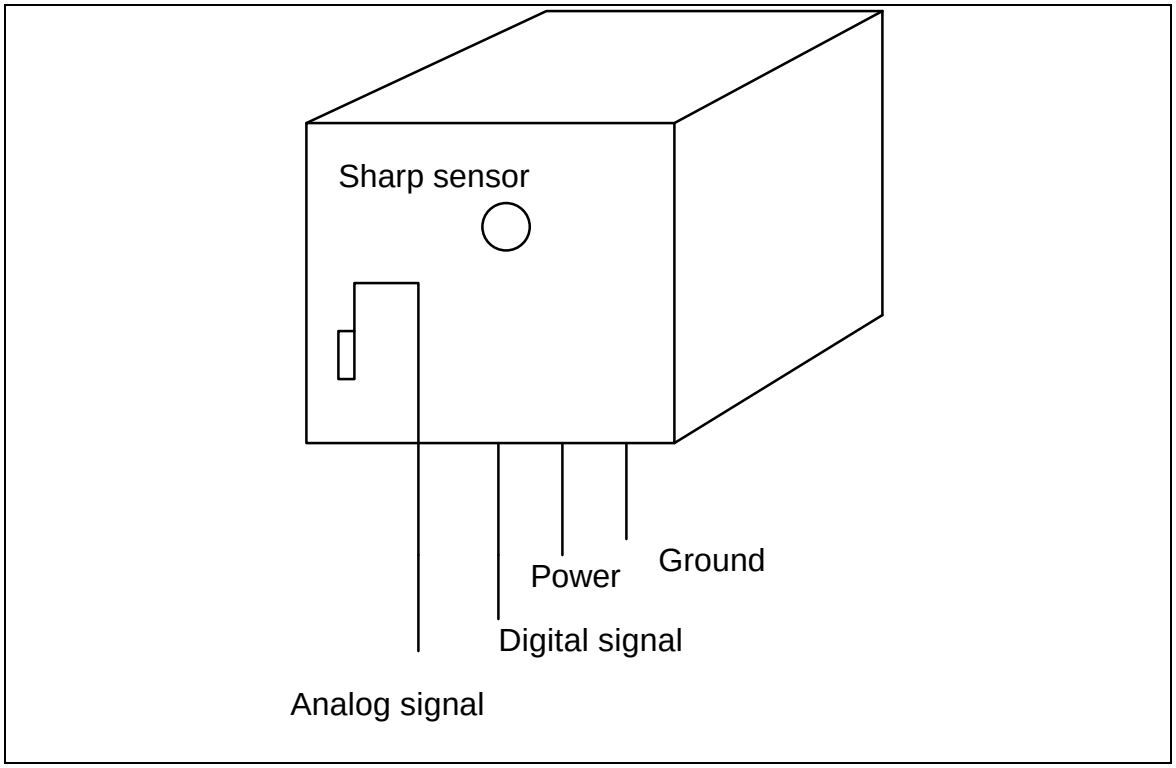

Fig. 6- Modification done on Sharp Sensor GP1U5 series 8X

* In addition to the modification shown above the case is also grounded.

The first experiment to attach a bank of five IR emitters in series was not successful. There was not enough current after the third IR sensor to drive any more emitters to produce stable results from the receivers.

I used memory map addressing as the final design to drive individual IR emitters. The receivers are all attached in series. Using a PAL VT220 and a latch I used the memory location $4000 to control the sensors.

Although there are only six sensors installed on Gopher, a maximum of eight different sensors can be accommodated using the present program of the PAL.

Wiring and connection of all these sensors to the Motorola board was another unexpected challenge. I used Kaynar wrapping wires to connect these sensors together and to the Motorola HC6811 board. Choosing this kind of wire as opposed to the thicker and perhaps more durable wire proved to be quite effective in making a bundle of about thirty sets of wires more manageable.

Fig 7 depicts the wiring schematic of the IR selection control circuit.

2 - ULTRASOUND SENSOR

The Third sensor that is used on Gopher is a pair of Ultrasound emitter/receiver sensor. This pair is mounted on the side of Gopher for the purpose of wall following behavior. Even though Wall following can be accomplished using just the infrared sensors, however it would be more appropriate for Gopher to detect distances of more than the 8" or 9" range that generally these Sharp IR sensors are capable of detecting. The distance recognition comes into play when Gopher is "faced" with an opening in the wall it has been following. This opening can be a legitimate path that Gopher needs to make a turn into or it may be a separation in the continuity of the object to its side. (separation between a row of boxes it has been following on it's side)

The transmitter of this circuit is designed to transmit a burst of 40khz signal with a peak voltage of 10 V. for a certain time period and be silenced (turned off) afterward. This allows the receiver to "listen" to the reflection of this sound for the silent period of the transmitter. Clocking the time it took the receiver to "hear" this reflected sound provides the information required to calculate the distance from the transmitter/receiver location to the object in front of them that the sound was bounced off.

At the time of this report there is a problem on the receiver end of the ultrasound sensory circuit. The reflected 40khz signal at the first stage of the OP AMP (receiver side) is changed to a sinusoidal signal, as it is supposed to be. However this signal is lost at the second stage. Work on this circuit will be continued in the following semester.

Fig 8A and fig. 8B depict the present wiring schematic of the Ultrasound receiver and transmitter circuits respectively.

Components used in implementing the sensors are listed in table 3.

Quantity	Description
5	Sharp GP1U5 series 8X
5	Infrared emitters
1	33K resister
2	Ultrasound 40K emitters
2	Ultrasound 40K receivers
1	74HC04 Hex Inverter
2	.01 Bypass capacitors
5	Shelf holder pegs as Sharp IR stands

Table -3 Components used in sensor implementation

Behaviors

The behavior realized using the present sensors on the Gopher are :

- **Straight line travel**
- **Object avoidance**
- **Determination of a "confused" behavior**
- **Think and escape behavior**

Straight line travel

This behavior is realized using the two miniature IR sensors embedded in the Futaba motors.

The first gear attached to the shaft of the Futaba motor assembly is painted into 4 quarters of black and white regions. There are two 5 V. pulses generated for each turn of this gear by the miniature IR sensor.

Since this wheel rotates once for each 41 rotations of the armature shaft, and the No. of pulses generated by the IR shaft encoder is determined by the Input Capture port of the Motorola board, the speed of the motor is therefore calculated.

See Appendix A for the algorithm used to test the shaft encoders.

Object avoidance

Object Avoidance being the most basic behavior is achieved by passing each IR signal (reflection of the infrared light from the object in front of the Sharp IR) through the A/D converter on the Motorola HC11 board. This is made possible because of the conversion made on the Sharp sensors to change their output signals into analog. (Sharp sensors are originally designed to produce digital signal)

Based on the intensity of the signal produced by the Sharp sensors the proper command is sent to the motors.

This information conveyed by this command contain: the specific motor No. to respond, the direction of its rotation, and the speed of its rotation.

Even though the concept of object avoidance is straight forward and simple; However there are different elements that contribute toward the development of a good or just an ordinary object avoidance behavior. Elements like program code, and the IR threshold settings.

In our class of eleven students there were eleven well built, “intelligent” robots (obviously no doubt left that I think Gopher is a well built and “intelligent” robot) and some even had sophisticated sensors like Pyroelectric (heat sensing sensor), or entertaining peripherals like speech synthesizers. However Gopher was by far the most praised robot in terms of it’s object avoidance behavior by professor Doty (class professor) and many of my classmates. They all considered it’s object avoidance behavior the best in our class so far, and yet Gopher has only the very basic sensors at this time.

The object avoidance behavior being the most visible behavior, in my opinion has also the most important role on how the robot is perceived. A simple robot with a good object avoidance routine can look more “intelligent” than a sophisticated one with sophisticated sensors that keeps running into the objects.

In order to achieve the optimum result in object avoidance for Gopher, different experiments using different algorithms and threshold settings for the on board IR sensors have been performed. Based on these experiments it was observed that use of long and elaborate subroutines that read the IRs again carry the penalty of time delay. This delay may not be crucial when stopping, starting, or backing up, but can make a smooth object avoidance into a clumsy brush or even worse, a collision with the object when an immediate and sharp turn is required. Therefore, in Gopher’s object avoidance routine the turns are executed by sending direct commands to the motors rather than calling subroutines and handing the control of the motors and IR sensors to those subroutines. There is one main loop in the object avoidance algorithm that continuously reads the IR sensors. Based on the result of those readings the proper commands to the motors are executed. Another point in achievement of the favorable result in the object avoidance

algorithm was the high threshold settings of the IR sensor readings. These settings were experimentally established for each of the five IR sensors mounted on Gopher. The final lesson learned from using different object avoidance algorithms was that the only sensor reading needed to make a decision to turn right or left is the readings from the corner IR sensors. Involving the middle sensor or any other sensors for that matter in the decision making process, at least for the immediate left or right turns will result in the shy behavior that is not a preferred characteristic in situations like negotiating tight passages similar to passing through a tunnel.

Determination of a "confused" behavior

An undesirable behavior that is very common is the continuous back and forth motion of the robot that make the robot seem to be "stuck in a loop". By incrementing a variable representing the No. of consecutive times the robot has made clockwise and counter clockwise turns without going straight or back Gopher can determine if it is stuck in a corner or not. This variable, called CONFUSED in Gopher's program, is set to zero every time Gopher uses the Straight or Backup subroutine (goes straight or backs up), and is incremented for turns. Therefor consecutive turns will cause this variable to reach a predetermined maximum. When this maximum No. is reached the robot "knows" it is "stuck in a loop" and the necessary course of action provided by the program can be taken.

Think and escape behavior

Once Gopher "realizes" it is moving a lot and going nowhere, it will use the Think routine. This routine consists of a 4 second wait followed by a counter clockwise turn of about 270 degrees. Why 270 degrees? In fact any movement that will face the robot in a different direction than what it is facing will be helpful, but experimentally 270 degrees put Gopher on the course closest to what it was before it got stuck in the loop; Yet different enough to avoid getting in the "confused" behavior again.

See Appendix B for Gopher's object avoidance routine.

CONCLUSION

Gopher, a Motorola 68HC11 based robot was built on a perforated aluminum platform with two Futaba SFP-S148 Servo motors. There is a free rotating wheel (caster) on the back of Gopher. As was mentioned before a Motorola 68HC11 board upgraded with 32K external RAM provides the computing power for Gopher. The factory installed MC68HC11E9FN chip on this board is replaced with another member of 68HC11 microprocessor family, namely the MC68HC11A1FN (referred to as simply the A1 chip). This chip contrary to the E9 chip does not contain the Buffulo program hence more memory is available for additional programming.

There are two IR sensors mounted internally in the Futaba motors serving as shaft encoders that provide speed calculation and comparison between the driving wheel of Gopher. These sensors provide the straight line travel behavior.

Five other IR sensors, three mounted in front and two in the back of Gopher provide object avoidance behavior. One Ultrasound sensor (still being worked on at the time of this report) is mounted on the side with room on the other side of Gopher for adding an additional Ultrasound in the near future.

The control system for the sensor selection is achieved by the use of a PAL (22V10Q) at address $4000. This eight bit memory mapped I/O address allows control of eight different sensors, six being utilized presently.

Limitations

Besides the Ultrasound sensor that is still being worked on, there are other limitations associated with the performance of Gopher. These limitation are categorized below.

Power

Power consumption by Gopher is considered better than the average with a operation range of between three and three and a half hour. The power failure comes rather unexpectedly in the last two or three minutes of the operation, and deteriorating rapidly. If equipped with power sensing sensors, efficiency in seeking the power source would have been a major consideration.

Object detection and avoidance

The limitation of the range and coverage of the Sharp IR sensors cause “blind spots” in front and rear of Gopher. In case of Gopher these spots are primarily in front and close to the side wheels. Additionally the IR light reflection from dark objects is not strong

enough to be detected by the Sharp sensors, therefore running into the brown plastic wall skirts is not uncommon.

performance

The cumulative error caused by the rotation of the wheels limits the robots straight line travel to a maximum of eight to ten feet at best (the best result achieved in our class) and in case of Gopher the maximum distance traveled straight is only four to five feet.

Reliability

Mechanical

Being built just for educational purposes, reliability was not considered a factor in evaluating the performance of Gopher. Nevertheless, I believe that the reliability of Futaba motors is the determining factor in the overall reliability of the system. The only test done on the Futaba motors was the one done by Scott, our T/A who ran a Futaba motor continuously in a forward and reverse motion for a period of 48 hours without any failure.

Structural

The perforated aluminum platform and metal and plastic screws used to connect the parts together (as opposed to super glue) has made Gopher sturdy enough to be carried three times a week back and forth to the lab without the fear of falling apart. (structural strength was one of the disadvantages of using Lego for the platform)

Things that I would do differently

The only thing that I wish I had done differently is using the on board PAL for more than just one byte of memory mapped I/O address. The PAL is a relatively expensive and powerful chip capable of replacing at least two other chips on the Motorola board (the AND gate and the Inverter).Efficient use of a PAL could result in more space on the Motorola board, less cost in construction of the robot, and perhaps a more esthetically desirable chip layout.

Suggestions for the course

I believe a general paper or file documenting the experience and the lessons learned by the previous classes would have helped me a lot in avoiding some of the mistakes in wiring and chip installations that I made Therefor the following is my contribution to that paper.

- Beware of the direction of the chips to be installed on the board (different chips facing different directions can cause confusion when trouble shooting the circuit).
- Connecting wires to the top of the board or the bottom of the board has different consequences that need to be evaluated in each different design.
- Choice of wires to connect the sensors to the board can make a difference in manageability of the wires and ultimately in ease of trouble shooting of the design.
- Advance knowledge of the alternative designs can allow for more efficient use of chips like PALs and therefore more efficient board space management.

Appendix A

Shaft encoder program

```
/* Encoders.c -- encoder code for ic for 6.270 board rev 2.2.  */
/* Shaft encoders on digital ports 0 and 1 should be used first
   because they are more efficient. */
/* Shaft encoders on digital ports 2 and 3 are less efficient and
   should only be used after 0 and 1 */
/* In order to use encoders, plug a digital shaft encoder into digital
   port 0 -> 3, and call enable_encoder() with the port number.
   Now you can call read_encoder() with the port number to read the
   number of shaft counter ticks since the last reset, and reset_encoder()
   to reset that number.  The counter overflows after around 32000 ticks */

void enable_encoder(int i)
{
  if(i==0)
    {
      bit_set(0x1022,1);
      port0_shaft_count=0;
    }
  else if(i==1)
    {
      bit_set(0x1022,2);
      port1_shaft_count=0;
    }
  else
    enable_cheesy_encoders(i);
}

void disable_encoder(int i)
{
  if(i==0)
    {
      bit_clear(0x1022,1);
    }
  else if(i==1)
    {
      bit_clear(0x1022,2);
    }
  else
    disable_cheesy_encoders(i);
}

void reset_encoder(int i)
{
  if(i==0)
    port0_shaft_count=0;
  else if(i==1)
    port1_shaft_count=0;
```

```
  else if(i==2)
    port2_shaft_count=0;
  else if(i==3)
    port3_shaft_count=0;
}

int read_encoder(int i)
{
  if(i==0)
    return(port0_shaft_count);
  else if(i==1)
    return(port1_shaft_count);
  else if(i==2)
    return(port2_shaft_count);
  else if(i==3)
    return(port3_shaft_count);
  return(0);
}
```

Appendix B

Object Avoidance program

```
  int IR1=0;  /* Initializing the IR sensors */
  int IR2=0;
  int IR3=0;
  int IR4=0;
  int IR5=0;
  int IR6=0;
  int IR7=0;

  int confused;            /* int. to keep track of No. back & forth move */
  int wheel_left=100;            /*  storing the speed of each wheel */
  int wheel_right=100;

  int left=0;     /* used in motor commands  */
  int right=1;

/* Turning each IR on for a short period, get a reading & turn it off. */

void Check_IR(int IR)
{
  if (IR == 1)
  {
    bit_set(0x4000,0b00000001);
    msleep(40L);
    IR1 = analog(1);
    bit_clear(0x4000,0b00000001);
  }
  else if (IR == 2)
  {
    bit_set(0x4000, 0b00000010);
    msleep(40L);
    IR2 = analog(2);
    bit_clear(0x4000,0b00000010);
  }
  else if (IR == 3)
  {
    bit_set(0x4000,0b00000100);
    msleep(40L);
    IR3 = analog(3);
    bit_clear(0x4000,0b00000100);
}
  else if (IR == 5)
  {
    bit_set(0x4000,0b00010000);
    msleep(40L);
    IR5 = analog(5);
    bit_clear(0x4000,0b00010000);
}
```

```
   else if (IR == 6)
   {
     bit_set(0x4000,0b00100000);
     msleep(40L);
     IR6 = analog(6);
     bit_clear(0x4000,0b00100000);
}
}

/* The Strait subroutine                */

void Strait()
{
   motor(left,wheel_left);
   motor(right,wheel_right);
   confused=0;
return;
}

void Backup()
{
  Check_IR(2);
  while (IR2 >= 125)
  {
    Check_IR(2);
    motor(left,-50);
    motor(right,-10);
  }

   confused=0;
return;
}
void Think()
{
    int N;
    motor(left,0);
    motor(right,0);
    msleep(3000L);

  Check_IR(5);
  Check_IR(6);
/*   while ( (IR5 <= 127) && (IR6 <= 127) )  */
  for (N=1; N< 40; N++)
  {
    Check_IR(5);
    Check_IR(6);
    motor(left,-20);
    motor(right,20);
  }

   confused=0;
return;
}
```

```
 void main()
{
  init_motors();
  msleep(4000L);
  while(1)
  {
    Check_IR(1);
    Check_IR(2);
    Check_IR(3);

    if((IR1<115) && (IR2<127) && (IR3<115))
    Strait();

    if (confused >= 6)
    Think();

    if((IR1 > 127) && (IR2 > 128) && (IR3<127))
    Think();

     else if ( (IR1 >= 110) && (confused < 6) )
/*           Turn_CW();     */
{   motor(left,50);
  motor(right,-50);
  confused +=1 ;
}
     else if( (IR3 >= 110) && (confused < 6) )
/*           Turn_CCW();       */
{
  motor(left,-50);
  motor(right,50);
  confused +=1 ;
}
    else if(IR2 >= 125)
{
    Backup();
    msleep(20L);
   }
}
return;
}
```

Appendix C

Factory supplied specifications