ME5643 Integrated Term Project:

IR-Guided Automated Factory Robot

Fall 2011

Rakshith Asokan, Rezwana Uddin

1 Project Motivation

The goal of our project is to provide a low-cost solution for directing automated robots around a factory. For small-scale industry with few workers our project makes it possible to rely on automation while limiting the overhead on the user side of transporting a vehicle or device from one station of a factory to another. With this in mind, we have designed a system whereby users are able to learn from a device which station of the factory a robot is currently at, and from there make a decision as to where to send the robot next simply by inputting the next station to travel to. The guiding of the robot is offloaded to a base controller, and the robot is equipped with the proper logic to follow the guidance correctly. To enable guidance, this project relies on IR emitters placed directly on the factory floor, while the robot uses IR receivers to detect the emitters and make appropriate decisions as to how to move. The use of sensors such as this allows a robot to travel to different stations in a factory throughout the day, based on user input, even along the same lines it may have used earlier to travel to a different station, because its travel will depend on which emitters are enabled. This way even in a small space one can allow for numerous stations with very few emitter installations. Low-cost guidance systems such as this have many applications to industry, Automated Guided Vehicles (AGVs) are applicable and used currently in a wide variety of warehouse and factory settings.

2 Previous Work

The modern day operation of a factory or warehouse often includes the use of an Automated Guided Vehicle. Through our project, our hope is to model, using Basic Stamp 2, an AGV and also to propose an IR-based guidance system.

The very first AGVs were based on wired sensors. In order to use this type of system the work environment would have to have wire tracks embedded approximately 1-inch into the factory floor. In order to make use of these, the vehicle would need to be equipped with sensors which look for and follow the magnetic field emanating from the wires. In contrast to this system, although embedding IR emitters into the floor of the factory also includes the expense of cutting into the factory floor it does not require nearly as much installation since there is no need for there to be a continuous strip of IRs, a few intermittent beacons are enough to allow the robot to learn where to change direction. This however depends on reliable control of the vehicles steering and navigation.

Another option currently used is to use guided tape. This is a system by which colored or magnetic strips of tape are placed along the factory floor, and the robot is equipped with sensors, such as light sensors, which allow it to follow a line of a particular color. Although this type of system solves the problem of needing to cut into the factory floor and is more cost efficient than a wired system, it only allows for certain set paths, and does not allow the robot to make efficient use of space. This limits the capabilities and usability of the vehicle for smaller industries.

One of the most advanced options for an AGV makes use of a combination of a variety of sensors, such as a laser range-finder and gyroscopes, and dynamically determines where it is, and shortest possible path it can take, while planning around any obstacles. This type of system is the most flexible and does not require any changes to the work setting. Furthermore, if an AGV relying on this system should fail, the effect on factory uptime is limited due to the fact that similar such vehicles can plan around a failed AGV. Due to limitations on time and the Basic Stamp 2's computational limitations we chose not to pursue this approach.

Although the concept of an AGV is not a new one (the first AGV was designed in 1953!) our hope is that our prototype will validate the possibility of using an IR-based guidance system that will provide a lower cost approach, requiring less installation overhead and more flexibility than a wired navigation system while also providing some sophistication, ease of use, and easy expandability through its base controller logic.

3 General Design

Our prototype design is built on top of the Parallax Boe-Bot which is controlled by the Basic Stamp 2 microcontroller. Our system relies on interfacing an intelligent base controller with the Boe-Bot-based AGV. In order to accomplish communication between these two components, we make use of Parallax 912 MHz transceivers. We have also designed a prototype factory floor in order to demonstrate the way in which IR emitters would potentially be placed; this is discussed further in Section 3, Base Construction. Rather than relying on a continuous strip of IR emitters, we utilize the emitters as sparsely as possible, placing them only wherever we may want to allow the vehicle to change direction or to stop. Stopping would occur at particular factory station. The base controller decides which IR paths to light while also providing a user interface in order to allow a factory manager to input which station he or she would like the robot to travel to while receiving any error messages, as well confirmation that the robot has indeed arrived at the transmitted station.

4 Bill of Materials

In the following table, we have recorded all the materials necessary to build our prototype:

S.No	Part Name	Quantity
1	Board of Education	2
2	Basic Stamp 2 Microcontroller	2
3	Parallax 912 MHz Transceiver	2
4	Ultrasonic Distance Sensor	1
5	3-pin male/male headers	2
6	Servo/LCD Extension cables	3
7	Parallax Continuous Servo	2
8	Speaker	1
9	Diodes	36
10	1 Kohm Resistor	12
11	220 ohm Resistor	3
12	LCD	1
13	Polymer Platform	1

5 Base Construction and Design

To construct a prototype factory floor we chose to use four stations at opposite corners and first designed a simple path that the robot could follow. Below is a diagram of the floor with the paths highlighted. However, please note that these paths are not actually visible on the factory floor. Instead there are just a few openings at important junctions where the robot will need to change direction.

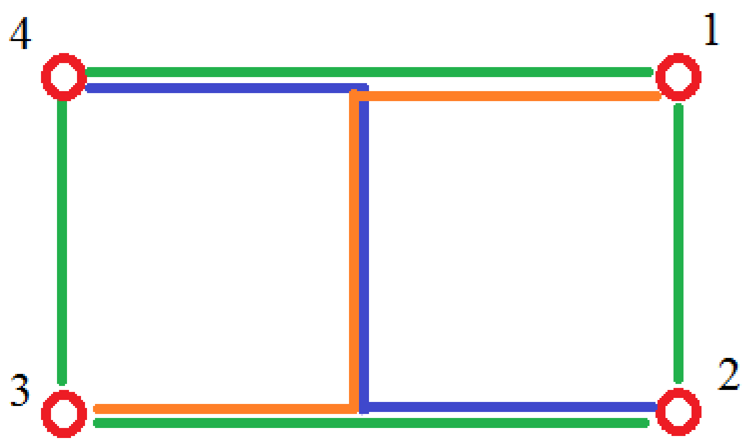

Fig 1. Robot Paths

The green paths are simply highlighting the paths the robot would follow if it were traveling to an adjacent station, for example, station 1 to 2, 2 to 3, 3 to 4, 3 to 2, etc. Although the orange and blue paths don't appear to overlap, the robot would actually be traveling along the same line down the middle when traveling from one corner station to another. In this way it is clear to see that we can reuse the same IR emitters to highlight different paths, and that the robot can travel again along a similar path so that space is used as efficiently as possible.

A circuit diagram of the board can be found in the Appendix as Circuit 1.

6 Robot Construction and Design

We built our prototype on top of the design for Parallax's Boe-Bot. After following the Boe-Bot instructions we included three IR detectors beneath the robot in order to allow it to detect a left emitter, center emitter, and right emitter at each junction. If the left junction were lit then the robot would know to turn left, if the center emitter were lit then the robot would stop, and if the right emitter were lit then the robot would turn right. Also if the robot started in a stopped position then in order to allow it to begin moving, both the center and another of the emitters would be lit. We also included an ultra-sonic detector with the intention to allow the robot to safely stop should an obstacle be detected. An LCD screen is attached in order to allow factory workers to know where the robot is headed as well as what it is currently doing, i.e. “Turning Left”, “Moving Forward”, etc.

The circuit diagram for the robot can be found in the Appendix as Circuit 2.

7 Programming

In the following sections we include a discussion of the programming being used to control our system. The programs themselves can be found in the Appendix.

7.1 Robot

The first thing the robot does is send using its wireless transceiver a signal to base indicating its current station. It then waits at a station until it detects an IR signal directing it to move.

In order to control the robot movement, the programming depends on reading the detector inputs simultaneously and checking whether the robot should move left, right, forward, or if it should stop.

Because of this, there are really only four cases necessary for the robot to check, depending on the detectors. There is also an additional case for when the detectors do not detect any input. Because we are not using a continuous strip of IRs the robot must check its previous state. If it was moving already, then it continues to move even if it does not detect any input. If it was stopped, then it remains stopped.

7.2 Base

The base controller logic is simplified by the design of the base circuit. Once the current station is received from the robot, the base allows the user to select the next station to send the robot to. It then decides which path to light up, indicated by the cases shown in the program. The program sends a FREQOUT command to light up the appropriate IR emitters.

8 Cost Analysis

8.1 Cost of One Unit

In the following table we summarize the cost of building one prototype AGV:

Table 1: Cost to produce 1 AGV

S.No	Part Name	Quantity	Cost/Part ($)	Total Cost ($)
1	Board of Education	2	69.99	139.98
2	Basic Stamp 2 Microcontroller	2	49.99	59.98
3	Parallax 912 MHz Transceiver	2	24.99	49.98
4	Ultrasonic Distance Sensor	1	19.99	19.99
5	3-pin male/male headers	3	0.50	1.50
6	Serco/LCD Extension cables	3	2.25	6.75
7	Parallax continuous servo	2	12.99	25.98
8	Speaker	1	1.95	1.95
9	Diodes	36	0.12	4.32
10	1Kohm Resistor	12	0.20	2.20
11	220 ohm Resistor	3	0.20	0.60
12	LCD	1	29.99	29.99

13	Polymer Platform	1	9.99	9.99
Total				373.12

8.2 Mass Production

In the following table we summarize the cost of mass-producing the AGV prototype.

Table 2: Cost to mass produce AGV

S.No	Part Name	Quantity	Cost/Part ($)	Total Cost ($)
1	Board of Education	2	55.99	111.98
2	Basic Stamp 2 Microcontroller	2	25.48	50.96
3	Parallax 912 Mhz Transceiver	2	16.49	32.98
4	Ultrasonic Distance Sensor	1	19.99	19.98
5	3-pin male/male headers	3	0.40	1.20
6	Serco/LCD Extension cables	3	1.80	5.40
7	Parallax continuous servo	2	11.69	23.38
8	Speaker	1	1.95	1.95
9	Diodes	36	0.08	2.88
10	1Kohm Resistor	12	0.16	1.92

11	220 ohm Resistor	3	0.16	0.48
12	LCD	1	23.99	23.99
13	Polymer Platform	1	7.99	7.99
Total				285.09

8.3 Cost Discussion

Cost analysis to produce one Prototype

Electronic components when purchased in bulk are cheaper than when sourced individually. From the table 1, the total cost of producing one robot is $373.11 .

Cost analysis for mass production

The cost from table 2 is $285.09.

Cost Comparison

Theoretically, the savings when a product is mass produced is from 25% to 30%.

Cost saved in mass producing the AGV is 373.11-285.09 = $88.02

Percentage savings is (88.02/373.11)*100 = 23.59%

Hence the percentage savings is close to the theoretical value of 25% and hence mass production of the AGV would be feasible.

9 Appendix

9.1 Programs

Robot Program:

```
' {$STAMP BS2}
' {$PBASIC 2.5}

NE VAR Nib
SE VAR Nib
NW VAR Nib
SW VAR Nib

base_confirm VAR Bit
base_confirm=0

RobotIN PIN 1
RobotOUT PIN 0

LMotor PIN 15
RMotor PIN 13

time VAR Word

curr_station VAR Nib
next_station VAR Nib

center VAR Bit
left VAR Bit
right VAR Bit

detect VAR Nib
old_detect VAR Nib

old_detect=%1110

pulseleft VAR Word
pulseright VAR Word

NE=1
SE=2
NW=4
SW=3
```

```
curr_station=NE
next_station=0
detect=%0000

robot_state VAR Byte
robot_state=0

'0=stopped
'1=moving
'2=turning left
'3=turning right

Main:
 base_confirm=0
 next_station=0
 DEBUG "Sending current station...", DEC curr_station, CR

  'Send out current station to base
  resend:
    SEROUT RobotOUT, 84, [curr_station]

  'Wait for a command from base to move to another station
  SEROUT 3, 84, [22, 12]
  SEROUT 3, 84, [DEC1 curr_station, 13, "Stopped"]
  DEBUG "Waiting for next station..", CR
  DO
    SERIN RobotIN, 84, [next_station]
  LOOP UNTIL next_station>0

  DEBUG "Got next station = ", DEC next_station, CR
  'Let base station know message was received
  SEROUT RobotOUT, 84, [next_station]

  'Next station received so move to it accordingly:
  'Check IR receivers to decide which way to move

DEBUG CLS
DO
    PAUSE 10
    GOSUB Check_Detect
    IF (detect=0 OR detect=4) THEN
      detect=old_detect
    ELSE
      old_detect=detect
    ENDIF

    IF time<=200 THEN
        pulseleft=750
        pulseright=770
        SEROUT 3, 84, ["Obstacle Detected"]
        GOTO Move
    ENDIF

    SELECT detect
      CASE 2, 8
         pulseleft=768
         pulseright=731
         robot_state=1
         DEBUG "Moving forward", CR
         SEROUT 3, 84, [DEC1 next_station, 13, "Moving forward"]
      CASE 14
         IF robot_state>0 THEN
           pulseleft=768
           pulseright=731
           'SEROUT 3, 84, [22, 12]
           'SEROUT 3, 84, [DEC1 next_station, 13, "Moving forward"]
         ENDIF
      CASE 12
        'turn left
        pulseleft=731
        pulseright=731
        robot_state=2
        SEROUT 3, 84, [DEC1 next_station, 13, "Turning left"]
        DEBUG "Turning left", CR
      CASE 6

        'turn right
         pulseleft=768
         pulseright=768
         robot_state=3
         DEBUG "Turning right", CR
         'SEROUT 3, 84, [22, 12]
         SEROUT 3, 84, [DEC1 next_station, 13, "Turning right"]
      CASE 10
         pulseleft=750
         pulseright=750
        IF robot_state>0 THEN
```

```
          'SEROUT 3, 84, [22, 12]
          SEROUT 3, 84, [DEC1 next_station, 13, "Stopping..."]
        ELSE
          SEROUT 3, 84, [22, 12]
          SEROUT 3, 84, [DEC1 next_station, 13, "Stopped"]
        ENDIF
         robot_state=0
         DEBUG "Stopping", CR
         curr_station=next_station
         GOTO Arrived
      ENDSELECT
    PAUSE 10

    Move:
    PULSOUT LMotor, pulseleft
    PULSOUT RMotor, pulseright
LOOP

'Added Check_Detect sub-routine
Check_Detect:
  PAUSE 20
  detect=INC
  detect.BIT0=0
  DEBUG "Check Detect: ", BIN4 detect, CR
  'Ultra-sonic
  PULSOUT 4, 5
  PULSIN 4, 1, time
  RETURN
Arrived:
DEBUG "Sending..", CR
DO
  SERIN RobotIN, 84, 20, here, [base_confirm]
  here:
  SEROUT RobotOUT, 84, [curr_station]
LOOP UNTIL base_confirm=1
SEROUT 3, 84, [22, 12]
SEROUT 3, 84, [DEC1 next_station, 13, "Stopped"]
DEBUG "Sent!", CR
RETURN

GOTO Main
```

Base Program:

```
' {$STAMP BS2}

' {$PBASIC 2.5}


NE VAR Nib

SE VAR Nib

NW VAR Nib

SW VAR Nib


BaseIN PIN 3

BaseOUT PIN 2


curr_station VAR Nib

next_station VAR Nib

robot_next VAR Nib

old_curr VAR Nib


i VAR Word


robot_confirm VAR Bit

robot_confirm=0


NE=1

SE=2

NW=4

SW=3
```

```
'Wait to get curr_station from robot
curr_station=0
old_curr=0

Main:

DEBUG "Waiting for current station.."

DO
  SERIN BaseIN, 84,[curr_station]
LOOP UNTIL NOT (curr_station=old_curr)

Send_Next:


DEBUG "Received: ", DEC curr_station, CR

'Enter next station for robot to move to
DEBUG "Enter next station: ", CR
DEBUG "NE=1", CR
DEBUG "SE=2", CR
DEBUG "SW=3", CR
DEBUG "NW=4", CR

DEBUGIN DEC1 next_station
```

```
'Send command to robot

SEROUT BaseOUT, 84, [next_station]


'Make sure robot received command

DO

  i=i+1

  SERIN BaseIN, 84, [robot_next]

LOOP UNTIL robot_next=next_station OR i>5000


'Light up the correct path based on where robot needs to go and from where it is
starting

SELECT next_station

  CASE NE

      IF curr_station=NE THEN

        DEBUG "You're already there!", CR


        GOTO Main

      ELSEIF curr_station=SE THEN

        GOTO SEtoNE

      ELSEIF curr_station=SW THEN

        GOTO SWtoNE

      ELSEIF curr_station=NW THEN

        GOTO NWtoNE

      ENDIF

  CASE NW

    IF curr_station=NW THEN

      DEBUG "You're already there!", CR
      GOTO Main
    ELSEIF curr_station=NE THEN
      GOTO NEtoNW
    ELSEIF curr_station=SE THEN
      GOTO SEtoNW
    ELSEIF curr_station=SW THEN
      GOTO SWtoNW
    ENDIF
  CASE SE
    IF curr_station=SE THEN
      DEBUG "You're already there!", CR
      GOTO Main
    ELSEIF curr_station=NE THEN
      GOTO NEtoSE
    ELSEIF curr_station=SW THEN
      GOTO SWtoSE
    ELSEIF curr_station=NW THEN
      GOTO NWtoSE
    ENDIF
  CASE SW
    IF curr_station=SW THEN
      DEBUG "You're already there!", CR
      GOTO Main
    ELSEIF curr_station=NE THEN
      GOTO NEtoSW
    ELSEIF curr_station=SE THEN
```

```
      GOTO SEtoSW
    ELSEIF curr_station=NW THEN
      GOTO NWtoSW
    ENDIF
  ENDSELECT

NEtoNW:
  DO

    FREQOUT 5, 5, 38500
    SERIN BaseIN, 84, 20, NEtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  GOTO Confirm

NEtoSE:
  DO
    'Light up the correct path
    FREQOUT 4, 5, 38500
    SERIN BaseIN, 84, 20, NEtoSE, [curr_station]

  LOOP UNTIL curr_station=next_station '
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NEtoSW:
  DO
```

```
    FREQOUT 6, 5, 38500
    SERIN BaseIN, 84, 20, NEtoSW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SEtoNE:
  DO
    FREQOUT 7, 5, 38500
    SERIN BaseIN, 84, 20, SEtoNE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SEtoSW:
  DO
    FREQOUT 8, 5, 38500
    SERIN BaseIN, 84, 20, SEtoSW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SEtoNW:
  DO
    FREQOUT 9, 5, 38500
    SERIN BaseIN, 84, 20, SEtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SWtoSE:
  DO
    FREQOUT 10, 5, 38500
    SERIN BaseIN, 84, 20, SWtoSE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SWtoNW:
  DO
    FREQOUT 11, 5, 38500
    SERIN BaseIN, 84, 20, SWtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SWtoNE:
  DO
    FREQOUT 12, 5, 38500
    SERIN BaseIN, 84, 20, SWtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoNE:
  DO
    FREQOUT 13, 5, 38500
    SERIN BaseIN, 84, 20, NWtoNE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoSW:
  DO
    FREQOUT 14, 5, 38500
    SERIN BaseIN, 84, 20, NWtoSW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoSE:
  DO
    FREQOUT 15, 5, 38500
    SERIN BaseIN, 84, 20, NWtoSE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm
```

```
Confirm:
  old_curr=0
  SEROUT BaseOUT, 84, [1]
  DEBUG "Robot arrived at appropriate station", CR
  GOTO Send_Next
    FREQOUT 11, 5, 38500
    SERIN BaseIN, 84, 20, SWtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

SWtoNE:
  DO
    FREQOUT 12, 5, 38500
    SERIN BaseIN, 84, 20, SWtoNW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoNE:
  DO
    FREQOUT 13, 5, 38500
    SERIN BaseIN, 84, 20, NWtoNE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoSW:
  DO
    FREQOUT 14, 5, 38500
    SERIN BaseIN, 84, 20, NWtoSW, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm

NWtoSE:
  DO
    FREQOUT 15, 5, 38500
    SERIN BaseIN, 84, 20, NWtoSE, [curr_station]
  LOOP UNTIL curr_station=next_station
  DEBUG "Arrived at: ", DEC curr_station, CR
  GOTO Confirm



Confirm:
  old_curr=0
  SEROUT BaseOUT, 84, [1]
  DEBUG "Robot arrived at appropriate station", CR
  GOTO Send_Next
```

9.2 Circuits

The circuit diagrams appear on the following page.