Alex Silverman
TA: Scott Nortman
TA: Aamir Qaiyumi
A. A Arroyo

University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory
FINAL REPORT

TABLE OF CONTENTS

Opening

Abstract--Pg 3
Executive Summary--Pg 3
Introduction---Pg 4

Main Body

Integrated System--Pg 5
Mobile Platform---Pg 5
Actuation---Pg 6
Sensors---Pg 6
Behaviors---Pg 6

Closing

Conclusion--Pg 8
Documentation---Pg 9
Appendix (code)---Pg 10

OPENING

Abstract:

My robot, Obot-ray, was designed to be an autonomous competitor in a game of my creation. In this report, I will first discuss the actual game the robot was to participate in; followed by a discussion of Obot-ray.

The game was to be similar to a tank simulator. In the game, the field and obstacles, if any, were to be unbeknownst to the robot at the start of the match. Thus, using only real time information, the robot must navigate through the field, and combat with other competitors. In order to recognize the opponent, each competitor must have an IR beacon atop it.

Once the opponent is found, the robot must attack with a laser, aimed to hit a CdS cell. The laser is to fire over a one second period. Thus, in order to facilitate Robot's aim, it should slowly sweep the laser across the area were the opponent is.

The games to be played for this class will have one final twist. Obot-ray will not play against another robot, but against a human controlled machine. This provides for a way to interact with Obot-ray, and to compare his skill at the game to a control device that receives much more information at the game than it does, namely a human. Finally, the human controlled opponent was to be a turret with the actions controlled via a Nintendo controller.

Executive Summary:

In this section, I am going to talk about what actually worked throughout the entire process. To begin with, the project began with assembly of the MRC11 and MSX01 boards. These boards were to play host to the intelligence system, and sensory information for Obot-ray.

Next, I hacked servos to act as dc motors to move Obot-ray around. I attached an unhacked servo to mount an IR can atop. I changed the Talrik servo code, so that it would also control the bridge servo with the same function.

After the boards, I assembled the platform. The platform was a standard Talrik II, platform for the most part. Coming off the front of the body is a plank. At one end, is the magnifying glass, and at the other a CDS cell to detect the laser. The detection system was tested, and found that it produced numbers that were far apart for bright light and the laser. So, Obot-ray could detect laser shots that hit it.

The IR rangers made life easy. First, they did not need to be hacked. The only changes that were made, was the power and signal wires needed to switch positions when put on the header.

Firing the laser proved more difficult than I thought. I decided on a BJT to fire the laser. It worked about 75% of the time. Also, whenever Obot-ray was off, or not running a program, the laser would stay on.

So, it looked like I only had software left for the most parts. Then, my MRC11 crashed, and some chips on it burnt out over thanksgiving. After that, I had to make repairs so that I finally got it running programs again. However, after his burnt out, many of Obot-ray's periferals would not work on a regular basis, and that is where my project ended.

Introduction:

The subject of robotics is a fascinating area. Many a book is written each year on how to build them, how they affect modern life, and what we should expect in the near future. I, however, know what to expect in the near future, the need to build a robot for IMDL. So, my task began with what kind of “brain”. I choose the ever popular at U.F. 68HC11. Many students are tempted to use it because it is cheap and widely available, that is why I chose the 68’11.

However, I also feel the 68’11 is good processor with its major downfalls being and antiquated instruction set and not nearly enough registers. So, my goal became to have my robot play a simple game, laser tag, and see just how much juice I can squeeze from our dear Motorola microprocessor.

So, In the paper I talk about the different systems I am using, i.e. the platform, actuation, sensors and other hardware. In addition, I discuss the code that gives the robot its behaviors and allow it to react to the world. Lastly, I talk about what I actually accomplished, problems encountered, and future work.

MAIN BODY

Integrated System:

The brains of Robot, as previously stated, is a Motorola 68HC11 based board with 64k expanded ram. In addition, I am using this board in conjunction with the sensory expansion board; this provides the interface for the majority of interactions my robot has with the real world. The board provides connections for many IR detectors, motor drivers, and servo drivers to say the least.

Mobile Platform:

The body of my robot is to be similar, if not an unmodified Talrik II body. This body provides good reliability in its simplistic design, is easy to assemble and make addition to, and looks to be light, which will increase speed. This platform is already set to use the MRC11 microcontroller and MRSX01 sensory board from Mekatronix(both designed by Scott Jantz). Furthermore, Obot-ray will have a much more complex sensor array than the CdS, IR, and Bump sensors found on T II. A picture of Talrik II's front is seen in Fig. 1(from www.mekatronix.com).

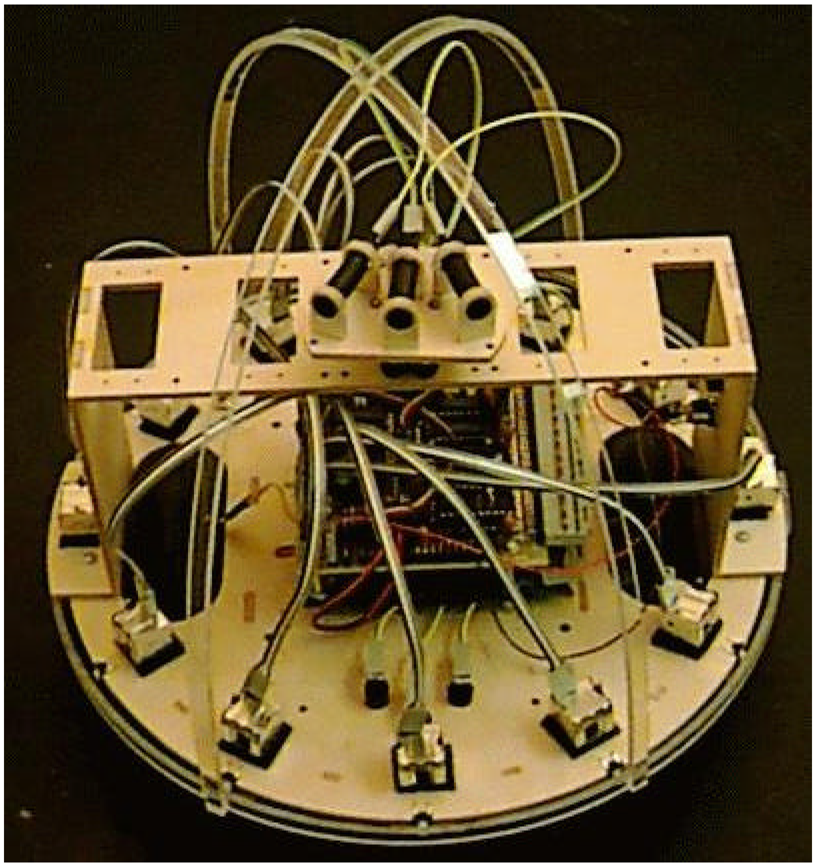

Fig. 1

Actuation:

Hacked servos propel Obot-ray about the playing field. While Servos do not provide the power(torque * speed) of some DC motors, they offer ease in that the controller circuitry for them is already built into them and the sensory board. To use the servos, I performed the Talrik hack described by Mekatronix in the Talrik assembly manual.

In addition, so that Obot-ray can look around for opponents with out actually moving, a servo has also been mounted on the bridge. OC1 also controls this servo, like the two hacked servos.

Sensors:

Obot-ray, has 40 kHz and 32kHz IR detectors to use for obstacle avoidance and location of other competitor, bump sensors in case of collision, and CdS cells to detect hits. For the IR detectors, I used two Sharp rangers, since we were not able to locate replacement cans for the sharps that can be hacked. However, the rangers provide much more consistent data for range finding.

I have used an unhacked IR can for opponent detection. Using the unhacked can, allows for opponent detection to be performed with different frequency IR than the rangers use. With this setup, opponent detection does not interfere with the rangers.

In order to register hits, the robot must have more than a simple CdS cell to register hits. With just a CdS cell, or even a few, the area the competitors need to hit would make it very hard to aim, unless they are in very close proximity. In order to get around this problem each hit zone will have a convex lens to hit. The purpose of the lens is to create a larger hit zone that will direct shots onto the CdS sensor.

The laser used is a 5mW class IIIa laser pointer. The switch was made to be always closed, via super glue, and the laser was hooked up across the emitter-collector junction of a BJT. Then a single form the the HC11 through an inverter, to the base of the BJT turns on and off the laser.

Behaviors:

At the beginning being able to play competitively against a human controlled opponent was to be the most difficult part of the task at hand. To begin with, as all robots must, Obot-ray can demonstrate obstacle avoidance. This is for obvious reasons. Creative use of obstacles by the robot was to be an important behavior, but after the board crashed time did not allow for it.

So, the behavior scheme that was finally implemented on Obot-ray is thus. First, drive, reacting to collisions and avoiding obstacles. Then, information from the sensor mounted on the bridge servo is used to determine if an enemy is present. If the enemy is present the robot goes into shoot mode. In addition, hits are detected in the main loop. At first I thought detection of hits would need to be inerupt driven, but as time was running out, and the shots were long enough, I had Obot-ray simply poll the CDS cell to detect hits. If a hit is detected, Obot-ray spins around.

When an opponent is detected we go into shoot mode. For the shooting behaiveor, is decided to go with a long shot over a couple short pulses. This makes for much easier detection. So, to shoot, the robot fires its laser while rotating at a direction oppisite to the bridge servoes position from center. If the bredge servo is centered, then Obot-ray rotates one direction then the other while shooting.

Originally, I was to use sensor information to determine the best method to try to defeat the opponent with. AI for Obot-ray was to be an integral part of a successful design in my opinion. This was the reason for my choice of the simple platform, and keeping the hardware side as easy as possible. However, after the MRC11 board crashed, and I basically had to build a robot in a week fancy AI and behaviors were not an option.

CLOSING

Conclusion:

Finally, I successfully built ran, destroyed, and rebuilt an autonomous agent capable of playing laser tag. After having to repair my MRC11 and rewrite my code, Obot-ray's internal running was kept very simple. However, I was satisfied, after taking into account everything that went wrong, with what I was able to get Obot-ray to do.

My work was had some very severe limitations to it. By design, my robot was to be simplistic from a hardware standpoint. This should have been accomplished, but with trying to repair Obot-ray after catastrophic failure, I would describe the hardware as anything but simplistic. Due, to the many setbacks I faced, my coding time was limited, and thus in my simple software is apparent another limitation of my work.

New advice for students to follow in IMDL, is nothing new. Start early, and don't expect anything to work right away. Always have replacement chips for your boards, you never know what is going to burn out or when. Finally, take a light course load with this class, and expect a lot of sleepless nights.

If I had the project to start over, I would do a few things differently. To begin with, I would back up my files very, very often; disk failures are bad. I would always have had every spare part imaginable for repairs. Lastly, I would have made the hit detection, interrupt driven with the IC functions of the HC11. Although, it worked fine polling the analog port, with the way the laser max out the CDS cell, I could have had the hits generate an interrupt (like it should be done, hits take effect right away).

Documentation

Mekatronix. “Talrik Assembly Manual.” Gainesville, Fl 1999.

APPENDIX

I)Modified servotk.c so that OC1 also controlls third servo from peizo pin.

```
/**********************************************************************
```

```
 * Title        servotk.c  (Servo Drivers for Talrik)                *
 * Programmer   Drew Bagnell, Talrik drivers                         *
 * Date         July, 1998                                           *
 * Version      1                                                    *
 *                                                                   *
 * Description: Servo drivers based on the TJ/TJP standard drivers    *
 *                                                                   *

*********************************************************************/


/*************************** Includes
*********************************/
#include <hc11.h>
#include <mil.h>
/*********************************************************************
**/


/*************************** Constants
********************************/
#define HALFPERIOD 20000         /* 50Hz refresh for each servo */
/*********************************************************************
**/


/************************** Prototypes
********************************/
#pragma interrupt_handler servo_hand
void init_servos(void);
void servo(int, int);
void servo_hand();
/*********************************************************************
**/


/**************************** Globals
*********************************/
unsigned int width[3];       /* width[] is the on-time for each servo*/
unsigned int current_width;
char signal_state, servomask[3];
/*********************************************************************
**/



void
init_servos(void)
/*********************************************************************
**
 * Function: Initializes all necessary variables for the servos.
*
 * Returns: None
*
 *
*
```

```
 * Inputs
*
 *   Parameters: None
*
 *   Globals:    None
*
 *   Registers:  None
*
 * Outputs
*
 *   Parameters: None
*
 *   Globals:   width[], signal_state, servomask[], current_width
*
 *   Registers: TOC1, PORTB, TMSK1, TCTL1
*
 * Functions called: None
*
 * Notes: None
*

***********************************************************************
*/
{

  INTR_OFF();

  //install interrupt handler on OC1
  *((void (**)())0xffe8)  = servo_hand;

  CLEAR_BIT(OC1M,0xff);          /* Interrupt will not affect OC pins
*/

  width[0] = width[1] =  width[2] = 0;  /* Motors start turned off */
  TOC1 = 0;
  signal_state = 0;
  current_width = 0;
  servomask[0] = 0x10;                  // modified servo masks oc4
  servomask[1] = 0x80;                  // oc1
  servomask[2] = 0x08;
  /* PA gates on  */
  SET_BIT(PACTL,0x88);                  /* SET PORTA, BIT7 (OC1) to
output */
  SET_BIT(TMSK1,0x80);                  /* Enable OC1 interrupt */
  INTR_ON();
}


void
servo(int index, int newwidth)
/**********************************************************************
**
 * Function: Sets one of the servos to the speed defined by newwidth
*
 * Returns: None
*
```

```
 *
*
 * Inputs
*
 *   Parameters: index, newwidth
*
 *   Globals:    None
*
 *   Registers:  None
*
 * Outputs
*
 *   Parameters: None
*
 *   Globals: width[index]
*
 *   Registers:  None
*
 * Functions called: None
*
 * Notes: None
*

**********************************************************************
*/
{
  asm("ldd %index \n"
      "aslb \n"
      "ldy #_width \n"
      "aby \n"
      "ldd %newwidth \n"
      "std 0,y");
}



void
servo_hand()
/*********************************************************************
**
 * Function: Interrupt handler for servo signals.
*
 *         Interrupt is OC1. Servo_1 is on PA4 and Servo_2 on PA7.    *
 * Returns: None
*
 *
*
 * Inputs
*
 *   Parameters: None
*
 *   Globals:    width[], signal_state, servomask[], current_width
*
 *   Registers:  None
*
 * Outputs
*
```

```
 *   Parameters: None
*
 *   Globals:    current_width
*
 *   Registers:  TOC1, PORTA, TFLG1
*
 * Functions called: None
*
 * Notes: None
*

***********************************************************************
*/
{
  char odd, index;
  unsigned int pwidth;

/*
   signal_state = 0  -> Turn on servo0
   signal_state = 1  -> Turn off servo0
   signal_state = 2  -> Turn on servo1
   signal_state = 3  -> Turn off servo1
*/

  signal_state &= 0x07;         /* Only use last 2 bits */

  if ((signal_state & 0x04))
      index = 2;
  else
      {
      index &= 2;
      index = (signal_state >> 1);  /* index references current servo
*/
      }

  odd = (signal_state & 0x01);

  pwidth = *(width+index);

  if ((pwidth == 0)&&(!(odd)))
  {
    TOC1 += HALFPERIOD;
    signal_state++;
  }
  else
  {
    if (odd)
      TOC1 += (HALFPERIOD - current_width);
    else
      TOC1 += pwidth;

    PORTA ^= *(servomask+index);
  }

  current_width = pwidth;
  signal_state++;
  if (signal_state == 6)
    signal_state = 0;
   CLEAR_FLAG(TFLG1,0x80);          /* Clear OC1I flag */
```

II)MAIN CODE

```
/**************************** Includes
*********************************/
```

```
#include <mytk.h>
```

```
#include <stdio.h>
```

```
/*********************** End of includes
*****************************/

/***PROTOS****/
void obstacle(void);
int oplook(void);
void shoot(void);
int hit(void);
void die(void);

  int i,IR_delay,servo3p;
  unsigned int temp,temp2,exits;


void main(void)
/****************************** Main
**********************************/
{

  init_analog();
  init_clocktk();
  init_serial();
  //init_motortk();
  init_servos();
  mux_sensor(0x0b); //cds2
temp=0;
exits=0;
servo3p=1000;
while(1)
  {
            wait(100);
            obstacle();
            if (oplook())
                  shoot();
            if (hit())
                  die();



  }

}

/*************************** End of Main
*****************************/





void obstacle(void)
/************************obstacle******************************/
{
      temp=analog(2);
      temp2=analog(3);
      wait(100);
      if (temp>100)           //left eye sees somthing
```

```
            if (temp2>100)  //both eyes see somthing
                  {
                  servo(0,4000);
                  servo(1,4000);
                  wait(400);
                  servo(0,2000);
                  servo(1,4000);
                  }
            else
                  {
                  servo(1,3000);          //swerve if only see with one
eye
                  wait(100);
                  servo(1,4000);
                  }
      else
            if (temp2>100)          //left see nothing check right
                  {
                  servo(0,3000);
                  wait(100);
                  servo(0,2000);
                  }
            else                    //see nothing drive starit
                  {
                  servo(0,2000);
                  servo(1,4000);
                  wait(100);
                  }
}
/***************END of OBSTACLE************************/



int oplook (void)
/***************oplook********************************/
{
      if (servo3p==1000)
            {
            exits=0;
            while(exits!=1)
                  {
                  if (analog(4)>120)
                        {
                        if (servo3p<3000)
                               servo3p=1000;
                        else servo3p=5000;
                        return 1;
                        }
                  else
                        servo3p+=500;
                  if (servo3p>=5000)
                        {
                        servo3p=5000;
                        return 0;
                        }
                  }
            }
            else
            {
            exits=0;
            while(exits!=1)
                  {
                  if (analog(4)>120)
                        {
                        if (servo3p<3000)
                               servo3p=1000;
                        else servo3p=5000;
                        return 1;
                        }
                  else
                        servo3p-=500;
                  if (servo3p<=1000)
                        {
                        servo3p=1000;
                        return 0;
                        }
                  }
            }
            return 0;
}


/********************END of oplook**********************/





void shoot(void)
/******************Shoot******************************/
{
      if (servo3p==1000)
            {
            PORTA &=0xBF;           //turn on laser
            servo(0,4000);
            servo(1,4000);
            wait(1000);                   //1 second shot
            servo(0,2000);
            servo(2,4000);
            PORTA |= 0x40;
            }
      else
            {
            PORTA &=0xBF;           //turn on laser
            servo(0,2000);
            servo(1,2000);
            wait(1000);                   //1 second shot
            servo(0,2000);
            servo(2,4000);
            PORTA |= 0x40;
            }
}
```

```
/*********END ofSHoot***********************/




int hit(void)
/****************hit************************/
{
      if (analog(0)<40)
            return 1;
      else
            return 0;
}
/********END ofHIt****************************/




void die(void)
/************die********************/
{
      servo(0,4000);
      servo(1,4000);
      while (1);
}
/***********END ofDIE*********************/
```