Jiggabot

University of Florida
EEL 5666
Intelligent Machine Design Lab

Student Name: Jerone Hammond

Date: April 25, 2001

Instructor: Dr. A. Arroyo

Table of Contents

Abstract

Jiggabot is an autonomous cleaning robot with voice recognition capabilities. It's behaviors are controlled by an Motorola 68HC11 micro-controller. There are a total of four different actuators that operate this robot. Two motors are used to navigate the robot, one servo is used to lower/raise the cleaning mechanism and the other is used to drive the cleaning brush.

Executive Summary

This robot is designed to search smooth surfaced floors for foreign debris and then clean it up. While doing this action it is able to follow walls and avoid any obstacle that it might encounter. This robot is controlled by an Motorola 68HC11 micro-controller which controls four motors, five IR receivers and emitters and six different switches. There are 2 front IR's for obstacle avoidance and 2 right side IR's for wall following. There is also an IR and receiver in front near the floor used in a break beam configuration used to detect objects. There are also 4 bump switches located around the robot used as a fail-safe in case the IR's miss an obstacle. This micro-controller is also used in conjunction with a circuit, which gives the robot voice recognition. It is possible to train this circuit up to 15 different commands in its current configuration, the maximum limitation of the circuit is 60 words. Under the current configuration of the robot the initiation command is the only word programmed, in order to increase the number of words the input port must be expanded. The robot initially follows a wall and when it encounters a wall directly in front it switches mode and tries to go in a zig-zag motion in order to clean the floor in an efficient manner.

Introduction

I was inspired to build this robot because of my military experience. In the military I served as an aircraft electrician. There were many regulations that we had to deal abide to when working on jet aircraft. One of the most boring things we had to do is called a F.O.D. (foreign object debris) walk. This task entailed walking up and down the whole flight line twice a day looking for any debris, which could cause harm to the jet aircraft engine. I hated doing this since it was so boring and I had always thought about building something that would accomplish this task for me.

Mobile Platform

The platform I will use will be a simple platform that I designed in Autocad. It is a basic design consisting of two front wheels and one rear support bracket that gives the robot the capability of turning in a complete circle. The platform resembles the basic layout for any typical car. In the rear half of the platform I cut out a section for the cleaning mechanism. In this area I built a boom that will rise and lower depending if the break beam detected any objects. I made this platform a little oversized in order to give plenty of room, which made placing the components easier, but on the other hand it made the robot less mobile. The robot had a more difficult time making turns next to walls than the typical round robot, which could swivel in a smaller radius of a circle.

Actuation

In my design I used a total of 4 motors. There will be one servo for each wheel on the robot. I am also using a servo to raise/lower the cleaning mechanism. There will also be a 3-volt dc motor that will run the sweeper brush.

I will control the servos used to moving the robot by using the output compare on the micro-controller. This will allow me to control the duty cycle of the pwm signal, which will regulate the speed of the motor. Limit switches will control the raising/lowering of the cleaning mechanism. This in turn will let me control the distance that the boom lowers or rises. Finally I controlled the 3-volt dc motor with a relay. This motor will have its own power supply, which is connected to a relay. This relay will be connected to one of the Port D pins on the TJPRO board. Whenever an object is found I will send a 5-volt signal to the relay which will connect the motor to power and drive the sweeper brush.

Electronics

The TJPRO board was the circuit of choice in my robot. The board was readily equipped with a Motorola 68HC11 micro-controller. This board was very user friendly and was setup nicely for a robot applications. It has the IR bus setup as well as the bus for the servos. The input and output ports were very easily read and many of the libraries were already established in C. This board gave me more time to work on the construction of my robot and the software rather than spending large amounts of time dealing with interfacing issues.

Sensors

Jiggabot has a total of four different types of sensors. First of all I am using IR sensors for obstacle avoidance and detecting objects. There are two IR emitters and detectors located at the upper front of the robot. These two sensors are for the obstacle avoidance. These sensors are hacked in order to output a analog voltage. The voltage output of the detectors is then converted into a hexadecimal number by the micro-controller. The values range from 80_{16} to 130_{16} where 80_{16} is no IR and 130_{16} is full IR. I also used IR sensors for object detection. In this case I placed an IR detector and emitter directly across from each other with about ten inches between them and collimated both the IR receiver and emitter. This gave me a very narrow range to detect any IR light. Both the receiver and emitter were placed in front of the robot as close to the ground as possible. This would allow the robot to maneuver around and be able to detect any objects that may cross its path. This is essentially like a large-scale optical switch or break beam.

In this robot I also implemented some bumper switches. There are a total of four switches strategically placed around the exterior of the robot. There are two in front and two in the rear. The two up front are mainly for backup in case the IR's don't see any obstacles. This can be a problem when there are black obstacles. The black surface seems to saturate the IR light and detection is not always accurate. If any of the bump switches are pressed it will make the robot either back up or pull forward depending on which switch it was.

I also implemented some limit switches for the cleaning mechanism. On this system there are two lever switches. One switch will detect when the cleaning mechanism has reached the maximum height and the other will detect when the cleaning mechanism has reached the floor. Both of these switches are hooked up into a resistive network which give different analog outputs depending on which switch is activated.

Special Sensor

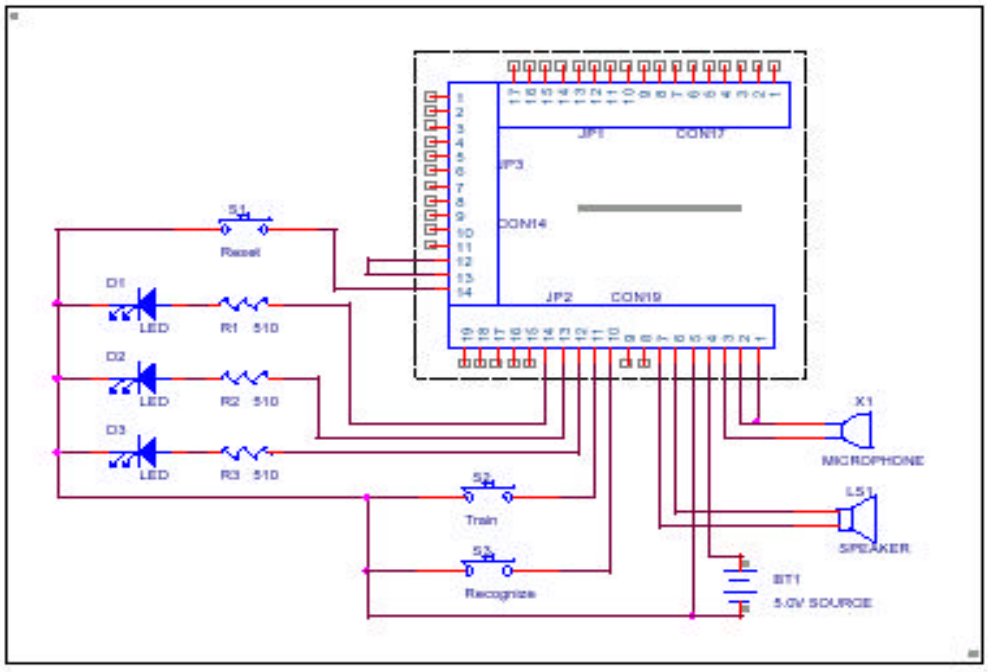
On this robot I decided to go with a voice recognition circuit for my special sensor. I was able to purchase this sensor pre-manufactured from www.VoiceActivation.com. This sensor's main component is a micro-controller in the RSC family, which a low cost 8-bit controller designed for use in consumer electronics. All members of the RSC family are fully integrated and include a speech processor, A/D, D/A, Rom and Ram circuitry on chip. The RSC family of micro-controllers can perform a full range of speech and audio functions including speech recognition, speaker verification, speech and music synthesis, and voice record and playback.

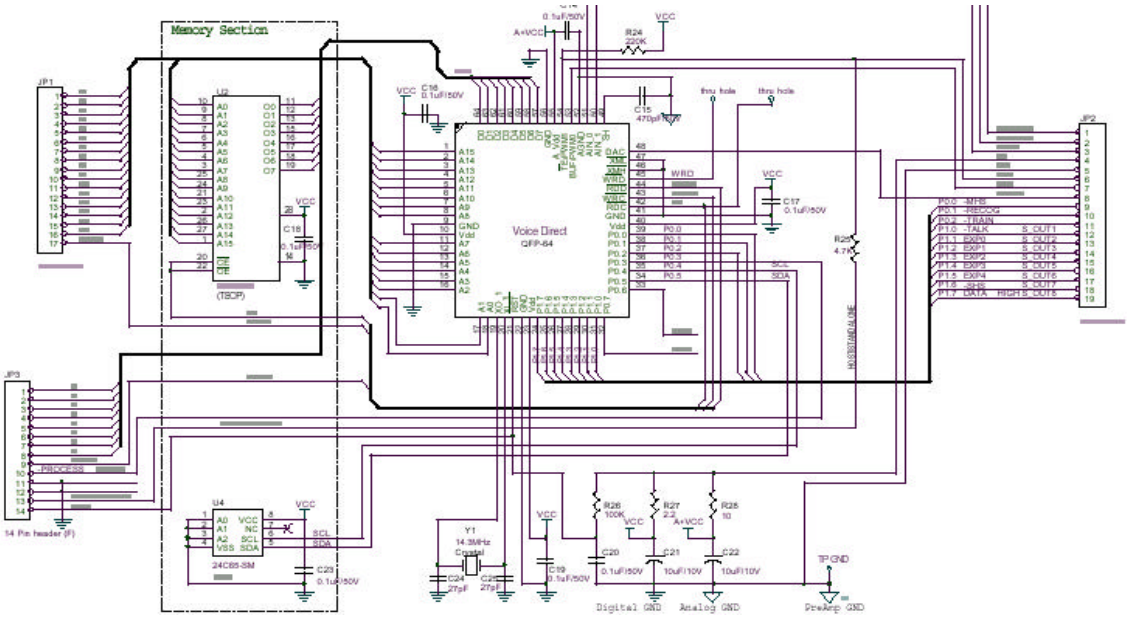
This particular sensor is capable of three different modes of operation: Speaker Dependent and two types of Continuous Listening. Speaker Dependent is initialized when a user presses a button then the senor will start listening for the

programmed commands. In Continuous Listening the user must program a key word plus the command word. The key word will signal the sensor to begin listening for the command word. The difference in both type of Continuous Listening mode is that one only has one keyword and the other has three different keywords. This sensor is capable of recognizing up to 60 words or phrases in slave mode or 15 words in stand alone mode. I will be using this sensor in stand alone mode. It is also capable of being 99% accurate depending on the proper design. The output that this sensor emits, are five volt pulses that stay high for one second at different pins according to the command word. The table below depicts the output compared to the command word.

Word 1	Output 1
Word 2	Output 2
Word 3	Output 3
Word 4	Output 4
Word 5	Output 5
Word 6	Output 6
Word 7	Output 7
Word 8	Output 8
Word 9	Output 8 and Output 1
Word 10	Output 8 and Output 2
Word 11	Output 8 and Output 3
Word 12	Output 8 and Output 4
Word 13	Output 8 and Output 5
Word 14	Output 8 and Output 6
Word 15	Output 8 and Output 7

The two figures below show the pin-out of the sensor and the schematic.



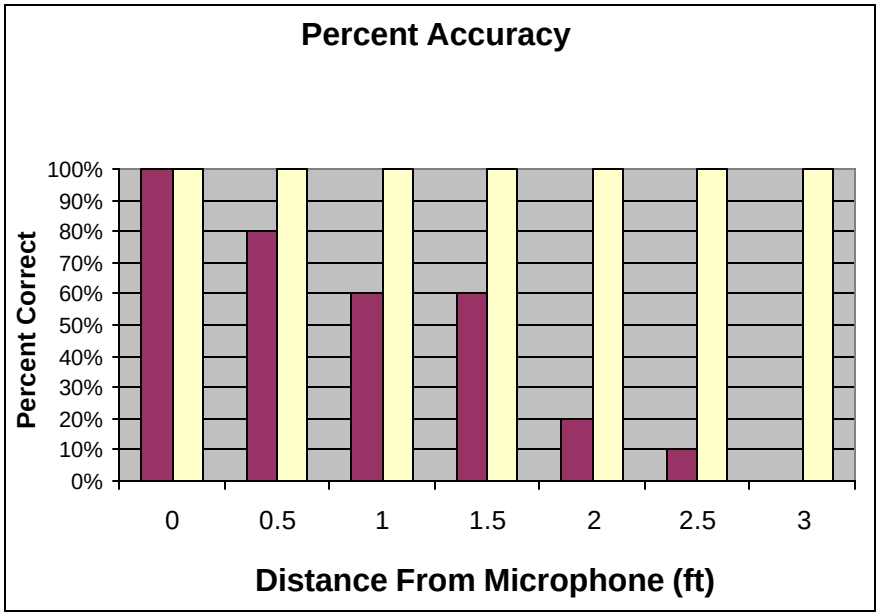


Behaviors

Jiggabot will have a simple set of behaviors. It will do nothing unless it is the given the voice command to do so. Once it is activated it will initially search for objects on the ground while following a wall. While it searches it will be avoiding obstacles that it might encounter. Eventually if it ever gets into a position where the front IR's get a certain specified distance from the wall it will go into just an obstacle avoidance mode. When it is in this mode it will try to search in a zig-zag pattern to effectively searching the entire area of the room. When it finally senses an object on the floor it will then stop and lower the cleaning mechanism. After it lower the cleaning mechanism the cleaning brush will start spinning and will then proceed forward at half speed. While it is cleaning it will proceed forward while still looking for objects on the ground. If there are no objects it will move forward a certain amount of time then stop and raise the cleaning mechanism into the upright position and then proceed forward at normal searching speed. Otherwise it will keep on cleaning.

Experimental Layout and Results

After assembling the voice recognition circuit I did a series of test to determine the accuracy of the circuit. In testing this circuit I did test measuring the accuracy compared to distance in a quite and noisy environment. In the graph below you can see that the accuracy drops tremendously as distance increases (red bar is amount correct). I repeated the test again with the servos running the results were even worse with the percent accuracy down to around zero. In order to alleviate the problem I tried to implement a wireless communication system by using a set of walkie-talkies. I was able to establish communications with the board but only at very short distances, even worse than with the existing microphone. I believe this was due to the inexpensive walkie-talkies that I purchased.



Conclusion

I have had this idea for this robot for a few years now. I felt that this project was fun as well as being somewhat challenging. I was able to successfully integrate the majority of the systems I wanted and the ones that were implemented were successful. Overall this robot was an accomplishment. This robot can be very useful in many different applications and could possible be a standard in the near future.

Appendix

Code for Jiggabot

```
/*****************************************************************************
*
 *                                                *
 * Title       jigabot.c                                *
 * Programmer  Jerone Hammond       *
 * Date        Spring 2001                          *
 * Version     1                                              *
 *                                                                  *
 * Description                                                      *
 *    A very simple collision avoidance program. TJ PRO will read  *
 *    each IR detectors, and turn away from any obstacles in its   *
 *    path. Also, if something hits TJ PRO's bumper, it will back up,*
 *    turn, and go on.                                             *
 *                                                  *
        *

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


/***************************** Includes *********************************/

#include <sensory2.h>
#include <stdio.h>
#include <hc11.h>
/************************* End of Includes ******************************/


/***************************** Constants ********************************/

#define AVOID_THRESHOLD 100
#define FAR   90
#define MODERATE 100
#define CLOSE         110

/************************* End of Constants *****************************/


/***************************** Prototypes *******************************/
void turn(void);
/************************* End of Prototypes ****************************/
```

```
void main(void)
/****************************** Main *********************************/
{
  int irdr, irdl, rfir, rbir, speedr, speedl;
  int mode = 0;
  int zigzag =0;
  init_analog();
  init_motortjp();
  init_clocktjp();
  init_servotjp();

  IRE_ON;     /* turn on IR emitters */


  START;  /*Press the break beam to start the program*/


   while(1)
  {

/*
   The following block will read the IR detectors, and decide whether TJ
   needs to turn to avoid any obstacles
*/

if(mode == 0)
{

        while ((Wall_FR < 95)&&(Wall_BA < 95)&&(RIGHT_IR < 90)&&(LEFT_IR <
90))
                {
                        motorp(RIGHT_MOTOR, SLIGHT_TURN);
                        motorp(LEFT_MOTOR, MAX_SPEED);
                }


        if((Wall_FR > Wall_BA )&&(Wall_FR > 95)&&(Wall_BA > 95))
                {
                   if (Wall_FR > CLOSE)
                        {
                                speedr = MAX_SPEED;
                                speedl = ZERO_SPEED;
                        }
                   else if ((Wall_FR > MODERATE) && (Wall_FR < CLOSE))
                        {
                                speedr = MAX_SPEED;
```

```
                        speedl = SLOW_TURN;
                }
            else if ((Wall_FR > FAR) && (Wall_FR < MODERATE))
                {
                        speedr = MAX_SPEED;
                        speedr = SLOWER_TURN;
                }
            else if (RIGHT_IR > CLOSE)
                {
                        speedr = MAX_SPEED;
                        speedl = SLOW_TURN;
                }

            else
                {
                        speedl = MAX_SPEED;
                        speedr = MAX_SPEED;
                }
     }

 else if ((Wall_BA > Wall_FR)&&(Wall_BA > 95)&&(Wall_FR > 95))
    {
            speedl = MAX_SPEED;
            speedr = 50;
    }

else if ((RIGHT_IR > CLOSE)&&(LEFT_IR > CLOSE))
                {
                        speedl = -MAX_SPEED;
                        speedr = -SLIGHT_TURN;
                        motorp(RIGHT_MOTOR, -SLIGHT_TURN);
                        motorp(LEFT_MOTOR, -MAX_SPEED);
                        wait(500);
                        mode = 1;
                }

      else if ((RIGHT_IR > MODERATE)&&(LEFT_IR > MODERATE))
                        {
                                speedl = ZERO_SPEED;
                                speedr = MAX_SPEED;
                                motorp(RIGHT_MOTOR, MAX_SPEED);
                                motorp(LEFT_MOTOR, -SLOW_TURN);
                                wait(400);
                        }

else
```

```
        {
                speedl = MAX_SPEED;
                speedr = MAX_SPEED;
        }

   motorp(RIGHT_MOTOR, speedr);
   motorp(LEFT_MOTOR, speedl);
   wait(70);
}

if(mode == 1)
{
        if ((LEFT_IR > 110)&&(RIGHT_IR < 110))
                {
                motorp(RIGHT_MOTOR, -MAX_SPEED);
                motorp(LEFT_MOTOR, MAX_SPEED);
        }
   else if ((RIGHT_IR > 110)&&(LEFT_IR < 110))
        {
                motorp(LEFT_MOTOR, -MAX_SPEED);
                motorp(RIGHT_MOTOR, MAX_SPEED);
        }

   else if ((LEFT_IR > AVOID_THRESHOLD)&&(RIGHT_IR >
AVOID_THRESHOLD)&&(zigzag ==0))
    {
        motorp(RIGHT_MOTOR, -MAX_SPEED);
        motorp(LEFT_MOTOR, MAX_SPEED);
        wait(700);
        motorp(RIGHT_MOTOR, MAX_SPEED);
        motorp(LEFT_MOTOR, MAX_SPEED);
        wait(400);
        motorp(RIGHT_MOTOR, -MAX_SPEED);
        motorp(LEFT_MOTOR, MAX_SPEED);
        wait(900);
        motorp(RIGHT_MOTOR, MAX_SPEED);
        motorp(LEFT_MOTOR, MAX_SPEED);
        zigzag = 1;
    }
    else if ((LEFT_IR > AVOID_THRESHOLD)&&(RIGHT_IR >
AVOID_THRESHOLD)&&(zigzag ==1))
         {
        motorp(RIGHT_MOTOR, MAX_SPEED);
        motorp(LEFT_MOTOR, -MAX_SPEED);
        wait(700);
        motorp(RIGHT_MOTOR, MAX_SPEED);
```

```
        motorp(LEFT_MOTOR, MAX_SPEED);
        wait(400);
        motorp(RIGHT_MOTOR, MAX_SPEED);
        motorp(LEFT_MOTOR, -MAX_SPEED);
        wait(900);
        motorp(RIGHT_MOTOR, MAX_SPEED);
        motorp(LEFT_MOTOR, MAX_SPEED);

        zigzag = 0;
         }



    else if((BUMPER > 20)&&(BUMPER < 30))
        {
                motorp(RIGHT_MOTOR, -SLOWER_TURN);
                motorp(LEFT_MOTOR, -MAX_SPEED);
                wait(1000);
        }
    else if ((BUMPER > 40)&&(BUMPER < 70))
        {
                motorp(RIGHT_MOTOR, -MAX_SPEED);
                motorp(LEFT_MOTOR, -SLOWER_TURN);
                wait(1000);
        }
    else if ((BUMPER > 70)&&(BUMPER < 120))
        {
                motorp(RIGHT_MOTOR, MAX_SPEED);
                motorp(LEFT_MOTOR, SLOW_TURN);
                wait(1000);
        }
    else if ((BUMPER > 120)&&(BUMPER < 140))
        {
                motorp(RIGHT_MOTOR, SLOWER_TURN);
                motorp(LEFT_MOTOR, MAX_SPEED);
                wait(1000);
        }
    else
        {
                motorp(RIGHT_MOTOR,MAX_SPEED);
                motorp(LEFT_MOTOR, MAX_SPEED);

        }
}
```

```
   if (Break_Beam < 90)
{
        motorp(RIGHT_MOTOR, ZERO_SPEED); /*Stop motors when object
detected*/
        motorp(LEFT_MOTOR, ZERO_SPEED);
        servo(1,4000);
        wait(1000);

        goto CLEAN;                                    /*go to cleaning subroutine*/
}



 CLEANBACK:
        wait(35);

 }

 CLEAN:
 {
        while (Switch > 80)
        {
                servo(1,4000);                  /*lower sweeper until switch
pressed*/
                }

        servo(1,0000);                  /*when switch presssed stop servo*/

        DDRD = 0x10;
   PORTD = 0x10;

        if (LEFT_IR > AVOID_THRESHOLD)
    speedr = -MAX_SPEED;
   else
    speedr = MAX_SPEED;
   if (RIGHT_IR > AVOID_THRESHOLD)
    speedl = -MAX_SPEED;
   else
    speedl = MAX_SPEED;

   motorp(RIGHT_MOTOR, speedr);
   motorp(LEFT_MOTOR, speedl);
```

```
  }

   wait(4000);

        if(Break_Beam > 90)
          {
                motorp(RIGHT_MOTOR, ZERO_SPEED); /*Stop motors when object
detected*/
                motorp(LEFT_MOTOR, ZERO_SPEED);
                servo(1,300);
                DDRD = 0x10;
        PORTD = 0x00;

                wait(5000);

        while (Switch < 130)
                {
                servo(1,300);
          }

                }
                servo(1,0);



        goto CLEANBACK;
}
/****************************** End of Main ******************************/

void turn(void)
/*************************************************************************
*
 * Function:  Will turn in a random direction for a "random" amount of  *
 *           time, dictated by the fast changine lower bits in           *
 *             mseconds().                                               *
 * Returns:  None                                                        *
 *                                                                       *
 * Inputs                                                                *
 *   Parameters: None                                                    *
 *   Globals:    None                                                    *
 *   Registers:  TCNT                                                    *
 * Outputs                                                               *
 *   Parameters: None                                                    *
 *   Globals:    None                                                    *
 *   Registers:  None                                                    *
 * Functions called: motorp(), wait()                                    *
```

```
 * Notes:                                                    *
*****************************************************************************
/
{
 int i;
 unsigned rand;

 rand = TCNT;

 if (rand & 0x0001)
 /*turn left*/
 {
   motorp(RIGHT_MOTOR, MAX_SPEED);
   motorp(LEFT_MOTOR, -MAX_SPEED);
 }
 else
 /*turn right*/
 {
   motorp(RIGHT_MOTOR, -MAX_SPEED);
   motorp(LEFT_MOTOR, MAX_SPEED);
 }

 i=(rand % 1024);
 if(i>250) wait(i); else wait(250);

}

/***********************End Function turn ****************************/
```

INCLUDE FILE SENSORY2.H

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

#include <analog.h>

#include <clocktjp.h>

#include <motortjp.h>

#include <servotjp.h>

#include <serialtp.h>

#include <isrdecl.h>

#include <vectors.h>

/************************* End of Includes *************************/

/************************** Constants ***************************/

#define LEFT_MOTOR        0

#define RIGHT_MOTOR     1

#define MAX_SPEED      100

#define ZERO_SPEED       0
#define SLOW_TURN                60
#define SLOWER_TURN              80


#define BUMPER     analog(0)


#define RIGHT_IR   analog(2)
```

```
#define LEFT_IR    analog(3)
#define Break_Beam analog(4)
#defineVoice1    analog(7)
#define Switch     analog(6)
#define Wall_FR          analog(5)
#define Wall_BA          analog(1)


#define START          while(Voice1<230)


/* Enable OC4 interrupt and all servo operations */
#define SERVOS_ON     SET_BIT(TMSK1,0x10)

/*Disable OC4 interrupt: Stops all servo holding torques, useful for energy savings*/
#define SERVOS_OFF    CLEAR_BIT(TMSK1, 0x10)

#define IRE_ON  *(unsigned char *)(0x7000) = 0x07
#define IRE_OFF       *(unsigned char *)(0x7000) = 0x00

/************************ End of Constants ****************************/
```

Jiggabot Photos

Error! Unknown switch argument.

Error! Unknown switch argument.

Error! Unknown switch argument.