

IMDL Final Report

EEL5666: Intelligent Machine Design Laboratory

Aaron Fisher

Robot:
Lazy Maid

Instructors:
Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

TA:
Thomas Vermeer
Mike Pridgen

Table of Contents

Abstract	3
Introduction.....	4
Integrated System.....	5
Mobile Platform	7
Actuation.....	9
Sensors	10
Behaviors	12
Experimental Layout and Results	14
Conclusion	16
Lazy Maid Code	17

Abstract

The Lazy Maid robot is a robot that is designed to clean a room. It will demonstrate this by picking up blocks and returning them to a bin in the corner of the room. The lazy maid has a mechanical arm mounted on the front which is what it will use to pick up and move the blocks. The CMU camera mounted below the mechanical arm is what the Lazy Maid uses to locate the blocks. The Lazy Maid uses IR sensors for obstacle avoidance and sonar sensors to determine how far away the block is. Finally, the Lazy Maid will be lazy because it uses a motion sensor to determine if anyone is in the room and will stop working if no one is there.

Introduction

The Intelligent Machines Design Lab requires each student to design and build an autonomous robot in a semester. I chose to design and build a cleaning robot that would pick up blocks and return them to a bin in the corner of the room. The robot was designed to stop working when there was no one in the room as well. This report details the design process as well as the results of designing this robot. The code used to perform the behaviors is attached to the end of this report.

Integrated System

The lazy maid uses the PVR board which was developed by the TAs in the Intelligent Machines Laboratory. This processor board was connected to two IR sensors, two sonar sensors, three bump switches and the CMU camera. These items were all essential to the operation of the robot.

The IR sensors shown in figure 1 are what the robot uses to perform obstacle avoidance. The two IR sensor attached to the top of the robot can be adjusted up or down for heightened sensitivity to either vertical objects or objects closer to the ground. Figure 1 also shows the bump switch just under the mechanical gripper. This bump switch triggers the gripper to close once the block is close enough for the robot to grab it.



Figure 1: Front of robot showing IR sensors

One of the bump switches is shown in figure 2. This switch is used to stop the robot from moving while the lid is open. Adding this bump switch has allowed me to have the robot on and fully programmed without it driving away while I am working on it.



Figure 2: The bump switch located under the lid

Figure 3 shows the CMU camera mounted just under the robotic arm. The camera was moved under the arm from on top of the robot to increase its visibility. Figure 3 also shows the two sonar sensors which are used to determine the position of the block relative to the robot. Finally, below the CMU camera there is a bump switch. This switch was used to determine if the robot was close enough to the bin to drop off the block.



Figure 3: The CMU camera mounted under the robotic arm

Mobile Platform

The Lazy Maid has a box on the back with a mechanical arm on the front. A full view of the Lazy Maid is shown in figure 4. The mechanical arm is a single degree of freedom arm which is made out of erector set pieces. The box on the back has a lid and no bottom to allow for easy access to from the top and bottom to the electronics inside.

The gripper on the end of the robotic arm was controlled by a servo pressing on the joint between the gripper pads. The gripper is held shut by a spring which provides two major advantages when compared to having the servo hold the gripper shut. First, the gripper will not require any power to keep the gripper closed. Second, the gripper can grip with far more power than was ever possible before the spring was added. Finally, gripper was equipped with large grain sandpaper glued to each pad. This was added from the previous report to increase the grip the robot could get on the block.

The robotic arm is controlled by a single servo at the base of the arm. Just behind this servo was a spring attached to the bend in the robotic arm which provided two benefits. First, the power used to get the servo to hold its position while holding a block was reduced. This was because the spring helped support the load while it was in the rest position. The next benefit was that the arm could now lift a heavier block than before. This ended up being essential to the success of the robot because the weight of the blocks kept them from sliding when the robot ran into them.

The robot was controlled with two servos, one on each side of the wood box. The robot was able to turn by spinning one servo one way and the other servo the other way. The wheel mounted just under the robotic arm supported the entire front of the robot. This wheel was a sphere with a hole through the center which allowed it to roll when going forward but slip when the robot turned from side to side.

Finally, the box on the back contained all of the electrical components for this robot. Hiding these components improved the visual appeal of the robot. The other major benefit of the large wooden box was counterbalance. The robotic arm and several sensors along with the block the arm was lifting provided a lot of weight on the front of the robot. The large wooden box over the main drive wheels kept the whole robot from tipping forward while the robot was operating.



Figure 4: Side view of the Lazy Maid platform

Actuation

The Lazy Maid operates off of four servos. Two of the servos are located on the mechanical arm and two of the servos are used to move the robot. The first servo on the mechanical arm was used to raise and lower the arm. The servo located further out on the arm was used to open the gripper. This servo only needs to open the gripper because the gripper uses a spring to come shut and clamp onto the block. The last two servos are connected to each of the wheels. The robot turns by simply rotating one wheel one way and the other wheel the other way.

The two servos on the arm were simple to control as they only have a few positions. The servo which controls the gripper was simply open or closed. A solenoid would have been ideal for this application however I had a servo and I would need to order a solenoid. Therefore, the servo was used to push the gripper open and closed.

The servo at the base of the robotic arm had three positions which it needed to achieve. These positions were arm up, arm down and arm drop off. The arm up was different then the position shown in the images in this report. The images in this report show the arm against the wooden box because the spring pulls it back that far while the servo is off. Arm down was when the gripper was placed just above the ground in a position to pick up the block. Arm drop off was used to place the block in the bin.

The two servos used to drive the robot were hacked servos so they could be used for continuous rotation. Each side behaved slightly differently due to differences in the servos so extensive testing was required for the drive system. By demo day the robot used five commands for the drive servos. These commands were hard right, hard left, medium right, medium left, medium forward. Hard right and hard left were used to when searching for the block and the bin. Medium left and medium right were used when aligning the robot with the block or the bin. Finally, medium forward was used when approaching the block or the bin. Medium forward was used rather the a full speed because it was more successful at making the robot go straight and it gave the sensors time to determine if the robot was aligned properly.

Sensors

The Lazy Maid uses four kinds of sensors. As shown in figure 1 the Lazy Maid uses IR sensors for obstacle avoidance and figure 3 shows the sonar sensors used for determining distance to the block. It also uses a CMU camera to determine where the block is so the robot can line up with it. Under the lid as shown in figure 2 there is a bump switch which is used to make sure the lid is closed before the robot goes about its behaviors. Finally, the Lazy Maid will use a motion sensor to determine if anyone is in the room. The motion sensor is not yet attached to the robot however.

The IR sensors were used to make sure the robot did not hit any objects while it was searching for blocks. Depending on the surroundings the IR sensors could either be tilted slightly up or slightly down. This allowed the IR sensors to detect either objects on the ground or more vertical objects such as a wall depending on how they were tilted. Each IR sensor was used simply to avoid obstacles and did not care how far away these obstacles were as would be needed in a function such as wall following.

The two sonar sensors were mounted on either side of the CMU camera. These sonar sensors would make sure the block did not go too far to the left or too far to the right. This was very important due to the size of the block compared to the opening of the gripper. Each block had to be within a couple of inches of directly in front of the robot otherwise the gripper would not align properly.

There were several bump switches used on this robot. The first bump switch was used under the lid of the wooden box. This bump switch simply determined if the lid was closed or open so that the robot did not move when the lid was open. The next bump switch was located under the gripper. This bump switch was hit when the block was in-between the gripper pads so that the robot would know to close the gripper. The final bump switch was located under the CMU camera. This bump switch told the robot that it was close enough to the bin to drip off the block.

The CMU camera was used to find the block. While the commands sent to the CMU camera were fairly simple it was highly sensitive to light and hence had to be recalibrated when moved from room to room. A camera shown in figure 5 was used to help the camera by controlling the lighting conditions as much as possible.

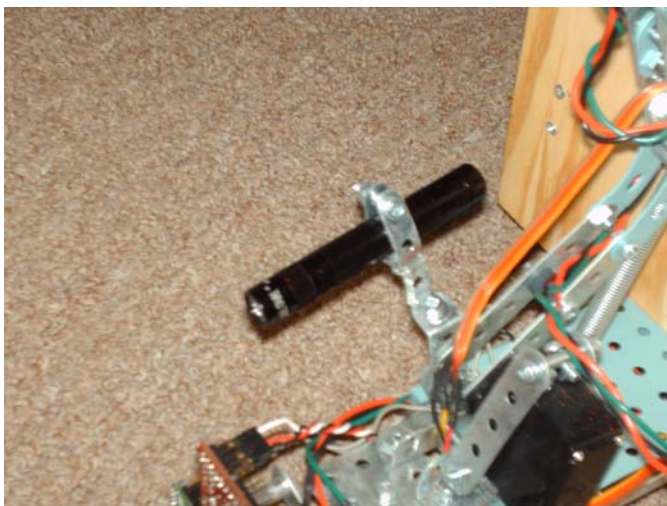


Figure 5: The light used to control lighting conditions

Lastly, this robot featured a motion sensor to determine if anyone was in the room. This motion sensor simply triggered when it sensed motion and did nothing when there was no motion. Each time the robot wished to detect motion it had to sit still for several seconds while the motion sensor established a baseline. Unfortunately, this sensor proved to be far too sensitive to demo during demo day.

Behaviors

The lazy maid will roam around the room looking for blocks to pick up and put away. Once the lazy maid finds a block it will align itself so that the mechanical arm can pick up the block. After the robot picks up the block it will return the block to a bucket located in the corner of the room. The lazy maid will continue to do this until it senses there is no one in the room and then it will go to sleep.

The blocks that the lazy maid was looking for are shown in figure 6. The color was chosen to be a bright orange color because it is very easy for the CMU camera to detect. The blocks are made with a wooden base and a foam piece sticking upward. Under the wooden base is a bit of rubbery gripping material. The piece sticking up was made of foam to make it easier for the robot to grip onto the block. The wooden bases were added so the blocks did not tip when the robot ran into them as well as weight so that the rubbery gripping material would keep the block from sliding on the floor.

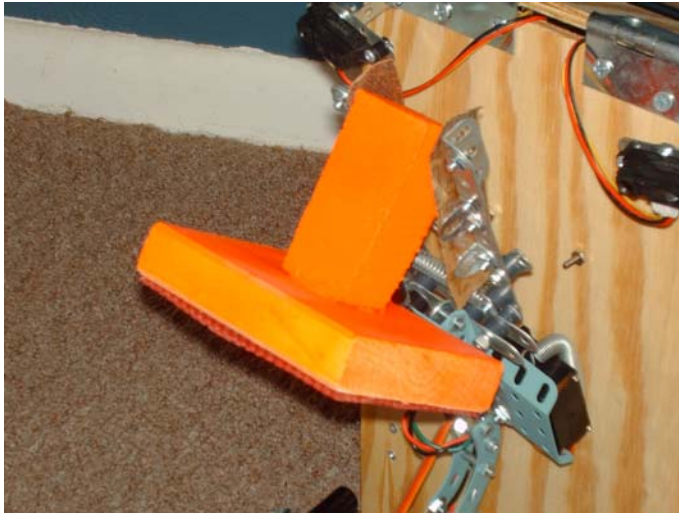


Figure 6: Shows the lazy maid holding one of the blocks

Once the robot picked up the block it then looked for the bin to put the block into. The bin was wrapped in green duct tape which was pick because it was significantly different then the orange used on the blocks. This was important so the robot did not confuse the blocks and the bin. Underneath the bin was the same rubbery gripping material used under the blocks so that the bin wound not slip when the robot bumped into it. The bin with several blocks in it is shown in figure 7.



Figure 7: The bin with two blocks in it

Experimental Layout and Results

The first thing that I experimented with was the IR sensors. To do this I simply had the LCD display show the value that it was reading for each sensor and I moved the robot towards a wall. I then took the values at several points and made a graph of the data. A picture of this is shown in figure 8 and the graph of the data for one of the IR sensors is shown in figure 9.



Figure 8: The test of the IR sensors

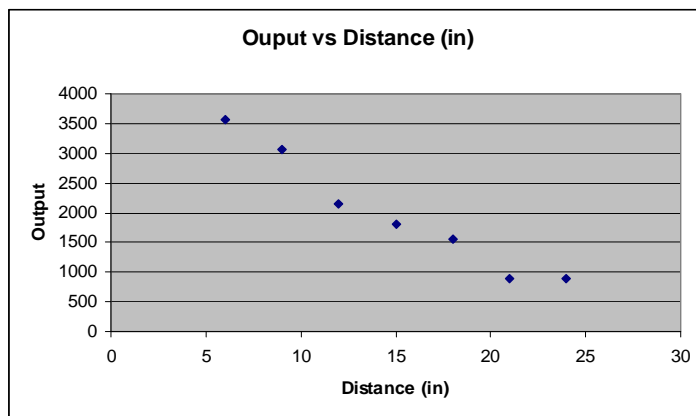


Figure 9: Output vs. distance for one for the IR sensors

While the graph above is taken from experimental data on this robot it will not always be accurate for this robot. This is because the IR sensors can be tilted up or down so their sensitivity to different kinds of objects might change. The bump switch was also tested by displaying its value to the LCD display but it only had two values, one value for open and one value for closed.

The next sensor that was tested was the CMU camera. This was done using the software that came with the CMU camera. I painted a stick the same color orange that was used on the blocks. A picture of that stick through the CMU camera is shown in figure 10.

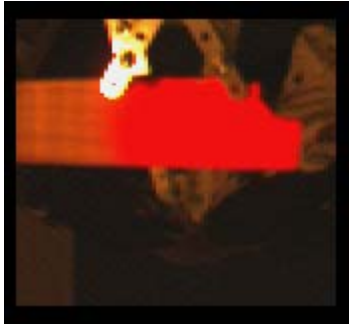


Figure 10: Orange stick viewed from CMU camera

The color values for the stick turned out to be roughly 240 red, 16 blue, 16 yellow. By using these numbers the camera tracks the stick very accurately which shows it will work well when locating blocks.

Unfortunately, despite several tests that were run on the CMU camera it is very sensitive to lighting conditions. A light was added to help with this problem but it was not completely eliminated.

The sonar sensors were tested by simply placing the block in an acceptable location in front of the robot and reading the values. Then the block was moved a little to the left and a little to the right. Using these values the robot knew rather it needed to move left, move right or attempt to pick up the block.

The last sensor that was tested was the motion sensor. This sensor was tested by first initializing it for the required time and then moving closer and closer to it until it triggered. Initially the sensor proved to be far too sensitive to any kind of motion but this improved as I blocked its view to a specific location. Unfortunately, the sensor was still too sensitive to demonstrate on demo day.

As for the servos on the lazy maid they were tested mostly using trial and error. Each value was programmed in until the robot moved straight or right or left or whatever was desired at the time.

Conclusion

The robot was mostly successful at performing the desired behaviors. The lighting conditions at the time of the demonstration caused the robot to not align itself properly. However, the robot did align itself properly in several other lighting conditions and hence was eventually demonstrated successfully. The only sensor that was never demonstrated was the motion sensor because there was far too much movement for it to work successfully.

The problems that were encountered during the design and building of the robot were due to the serial connection with the CMU camera, the blocks being quite small and equipment failure. The CMU camera was easy to communicate with once a baseline code was established. Unfortunately, I had no experience with serial communication before attempting this so progress was stalled for several weeks while I tried to learn all I could. I was able to successfully use the CMU camera in the end thanks to several of my classmates which are credited in the code section of this report.

The bin was easy for the robot to find and align with as it was pretty bit but the blocks were not so easy. Because of the size of the blocks and the size of the gripper if the blocks were one inch to the right or left the gripper would miss. Therefore, aligning the robot was a significant challenge. In the end I used the sonar sensors to get the robot close to aligned with the block and then I ran into the block with the gripper open. This helped guide the block into the gripper and so the robot did not have to be as accurate when aligning with the block.

Finally, equipment failing near the end of the course was a large cause of problems. Both drive servos stopped working and one of the bump switches started to fail. I believe the problem with the servos was I tried to get away with cheap servos ordered off of the internet and by the end of the semester they had reached the end of their life span. The bump switch failed but that was an easy and cheap fix once I realized what was wrong.

While I do believe there is more work to be done to make this robot perform better I am satisfied with the result after a single semester. I plan on continuing to work on the code and I plan on adding new features so that I have a very impressive robot to show on my website as well as to potential employers.

Lazy Maid Code

Code written to control the servos

```
#include <avr/io.h>
#include "servocontrol.h"
#include "PVR.h"

void Lidopen(void)
{
    TCC0_CCA = 0; //PWMC0 off

    TCD0_CCA = 0; //PWMD0 off
}

void Stop(void)
{
    TCC0_CCA = 0; //PWMC0 off

    TCD0_CCA = 0; //PWMD0 off
}

void FF(void)
{
    ServoC0(100);
    ServoD0(-100);
}

void MF(void)
{
    ServoC0(-100);
    ServoD0(-55);
}

void SF(void)
{
```

```
ServoC0(83);  
ServoD0(-10);
```

```
}
```

```
void Rev(void)  
{
```

```
ServoC0(100);  
ServoD0(100);
```

```
}
```

```
void HR(void)  
{
```

```
ServoC0(100);  
ServoD0(-20);
```

```
}
```

```
void MR(void)  
{
```

```
ServoC0(-5);  
ServoD0(-50);
```

```
}
```

```
void SR(void)  
{
```

```
ServoC0(70);  
ServoD0(-50);
```

```
}
```

```
void HL(void)  
{
```

```
ServoC0(-100);  
ServoD0(20);
```

```
}
```

```
void ML(void)  
{
```

```
ServoC0(-100);  
ServoD0(-10);
```

```

}

void SL(void)
{

ServoC0(100);
ServoD0(20);

}

void Armdown(void)
{

ServoD5(20);

}

void Armup(void)
{

ServoD5(-85);

}

void Armdrop(void)
{

ServoD5(-50);

}

void Gripperopen(void)
{

ServoD1(100);

}

void Gripperclose(void)
{

ServoD1(-100);

}

```

Code written to collect the data from each of the sensors

```

#include <avr/io.h>
#include "PVR.h"

```

```

int getdatacnt[10];
int ans;

unsigned int SonarR(void)
{

    int i=0;
    int cnt=0;
    while(i<11)
    {
        getdatacnt[i]=ADCA1();
        i++;
        delay_ms(30);
    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/10;
    delay_ms(10);
return ans;
}

```

```

unsigned int SonarL(void)
{

    int i=0;
    int cnt=0;
    while(i<11)
    {
        getdatacnt[i]=ADCA3();
        i++;
        delay_ms(30);
    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/10;
    delay_ms(10);
return ans;
}

```

```

unsigned int Blockbump(void)
{

```

```

    int i=0;
    int cnt=0;
    while(i<11)
    {
        getdatacnt[i]=ADCA7();
        i++;
        delay_ms(30);
    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/10;
    delay_ms(10);
return ans;
}

```

```

unsigned int Binbump(void)
{

```

```

    int i=0;
    int cnt=0;
    while(i<11)
    {
        getdatacnt[i]=ADCA6();
        i++;
        delay_ms(30);
    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/10;
    delay_ms(10);
return ans;
}

```

```

unsigned int Right(void)
{

```

```

    int i=0;
    int cnt=0;
    while(i<21)
    {
        getdatacnt[i]=ADCA2();
        i++;
    }

```

```

    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/20;
    delay_ms(100);
return ans;
}

```

```

unsigned int Left(void)
{

```

```

    int i=0;
    int cnt=0;
    while(i<21)
    {
        getdatacnt[i]=ADCA4();
        i++;
    }
    while (i>0)
    {
        cnt+= getdatacnt[i];
        --i;
    }
    ans=cnt/20;
    delay_ms(100);
return ans;
}

```

The code written to use the CMU camera (original was written by Josh Phillips)

```

#include <avr/io.h>
#include "uart.h"
#include "cmucam.h"
#include "xmega.h"

```

```

int getgreendata[7];
int getdatacnt[20];
int i;
int value[6];
int ans;

```

```

void CMUcamInit(void)
{

```

```

    uart_sendstring("RS\r") ;           //reset camera

```

```

delay_ms(5000);                                //give camera 5 seconds to reset itself

uart_sendstring("DM 64\r");                    //set a delay of 2 byte period for serial transfer

delay_ms(1000);                                //delay for feedback

}

void orange(void)
{

    uart_sendstring("NF 1\r");

    delay_ms(300);

    uart_sendstring("MM 1\r");
    delay_ms(300);

    uart_sendstring("TC 240 240 30 50 10 30\r");
    delay_ms(300);

}

void green(void)
{

    uart_sendstring("NF 1\r");

    delay_ms(300);

    uart_sendstring("MM 1\r");
    delay_ms(300);
    uart_sendstring("TC 70 80 190 250 10 70\r");
    delay_ms(300);

}

unsigned int getdata(void)
{

```

```

int i=0;
int cnt=0;
while(i<21)
{
getdatacnt[i]=uart_getint();
i++;
}
while (i>0)
{
cnt+= getdatacnt[i];
--i;
}
ans=cnt/20;
delay_ms(100);
return ans;
}

```

The PVR code and the UART library were also used for this robot. The following is the main code for the Lazy Maid robot.

```

#include <avr/io.h>
#include "PVR.h"
#include "uart.h"
#include "cmucam.h"
#include "servocontrol.h"

void main(void)
{
    xmegaInit();           //setup XMega
    delayInit();          //setup delay functions
    ServoCInit();         //setup PORTC Servos
    ServoDInit();         //setup PORTD Servos
    ADCAInit();           //setup PORTA analog readings
    lcdInit();            //setup LCD on PORTK
    lcdString("Starting up"); //display "PV Robotics" on top line (Line 0) of LCD
    uart_init();          //setup the uart directory

    CMUcamInit();        //setup the CMUcam
    int getorange;
    int getdist[5];
    int getsonarR;
    int getsonarL;
    int getorange2;
    int getorange3;
    int gr = 1;
    int getgreen;
    int turn = 3;
    int FF=0;
    int getirright;
    int getirleft;
}

```

```

while(1)
{
    lcdData(0x01);                //clear lcd

if (ADCA0(>500)
{
    Lidopen();
    lcdGoto(0,0);
    lcdString("Waiting for");
    lcdGoto(1,0);
    lcdString("Lid to Close");
    delay_ms(1000);
}
else
{

while((ADCA0(<500) && (gr=1))
{
    orange();
    lcdData(0x01);                //clear lcd
    HR();
    delay_ms(150);
    Stop();

    getorange = getdata();
    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Looking for");
    lcdGoto(1,0);
    lcdString("the block");
    delay_ms(1200);

    while (getorange>42)
    {
        Stop();
        lcdData(0x01);
        lcdGoto(0,0);
        lcdString("Found block");

        getsonarR = SonarR();
        getsonarL = SonarL();
        getorange2 = getdata();
        delay_ms(300);
    }
}
}
}

```

```

while ((turn = 3) && (getorange2<42))
{
HL();
delay_ms(150);
Stop();
delay_ms(1000);
getorange3 = getdata();
delay_ms(300);

if (getorange3>42)
{
turn = 4;
getorange2 = getdata();
delay_ms(300);
}
}

while (((turn = 4) && (getorange2<42)))
{
HR();
delay_ms(50);
Stop();
delay_ms(1000);
getorange3 = getdata();
delay_ms(300);
if (getorange3>=42)
{
turn = 3;
getorange2 = getdata();
delay_ms(300);
}
}

if(((getsonarR<380) || (getsonarL<380)) && ((getsonarR>365) ||
(getsonarL<356)))
{

getsonarR = SonarR();
getsonarL = SonarL();
getorange2= getdata();
lcdData(0x01);
lcdGoto(0,0);
lcdString("Aligning with");
lcdGoto(1,0);
lcdString("with the block");

if (((getsonarR<372) && (getsonarL>376)) && (getsonarR>365) &&
((getsonarR<500) || (getsonarL<500)))

```

```

        {
            MR();
            delay_ms(80);
            Stop();
            getsonarR = SonarR();
            getsonarL = SonarL();
        }
        else if (((getsonarR>372) && (getsonarL<376)) && (getsonarL>365)
&& ((getsonarR<500) || (getsonarL<500)))
        {
            HL();
            delay_ms(120);
            Stop();
            getsonarR = SonarR();
            getsonarL = SonarL();
        }
        else if (((getsonarR>372) && (getsonarL>376)) || ((getsonarR<365) &&
(getsonarL<365))) && ((getsonarR<500) || (getsonarL<500)))
        {
            MF();
            delay_ms(250);
            Stop();
            getsonarR = SonarR();
            getsonarL = SonarL();
        }
        else if ((getorange2>42) && (((getsonarR<=372) && (getsonarL<=376))
&& ((getsonarR>=365) && (getsonarL>=365))) && ((getsonarR<500) || (getsonarL<500)))
        {
            lcdData(0x01);
            lcdGoto(0,0);
            lcdString("Picking up");
            lcdGoto(1,0);
            lcdString("the block");
            Stop();
            delay_ms(500);
            Armup();
            delay_ms(500);
            Gripperopen();
            delay_ms(500);
            Stop();
            delay_ms(500);
            Stop();
            delay_ms(500);
            Armdown();
            delay_ms(500);
            FF=1;
            while ((Blockbump())>100)
            {
                if (FF<5)
                {
                    MF();

```

```

        delay_ms(80);
        Stop();
        delay_ms(200);
        FF++;
    }
    else if (FF>=5)
    {
        HL();
        delay_ms(60);
        Stop();
        FF=1;
    }
}
Stop();
delay_ms(600);
Rev();
delay_ms(50);
Stop();
delay_ms(600);
Gripperclose();
delay_ms(500);
Armup();
delay_ms(1000);
gr = 2;
getorange = 20;
delay_ms(200);

}
}

```

```

else if((getsonarR>=380) && (getsonarL>=375))
{
    getsonarR = SonarR();
    getsonarL = SonarL();
    MF();
    delay_ms(60);
    Stop();
    getorange2 = getdata();
    getsonarR = SonarR();
    getsonarL = SonarL();
    delay_ms(600);

```

```

        if ((getorange2<=42) && ((getsonarR>=380) &&
(getsonarL<=378)))
        {
            HL();
            delay_ms(120);
            Stop();
            getorange2 = getdata();

```

```

        getsonarR = SonarR();
        getsonarL = SonarL();
        }
        else if((getorange2<=42) && ((getsonarR<=378) &&
(getsonarL>=380)))
        {
        HR();
        delay_ms(60);
        Stop();
        getorange2 = getdata();
        getsonarR = SonarR();
        getsonarL = SonarL();
        }
    }
}

while ((ADCA0()<500) && (gr=2))
{
    green();
    lcdData(0x01);
    lcdGoto(0,0);
    lcdString("Looking for");
    lcdGoto(1,0);
    lcdString("the bin");
    HL();
    delay_ms(400);
    Stop();

    getgreen = getdata();

    delay_ms(1200);

    while ((getgreen>43)&&(gr=2))
    {

        lcdData(0x01);
        lcdGoto(0,0);
        lcdString("Found Bin");
        delay_ms(500);
        getsonarR = SonarR();
        getsonarL = SonarL();
        MF();
        delay_ms(300);
        Stop();
        delay_ms(200);
        if((Binbump()>3000))

```

```

    {
        Stop();
        delay_ms(300);
        Armdrop();
        delay_ms(300);
        Gripperopen();
        delay_ms(300);
        Armup();
        delay_ms(300);
        Gripperclose();
        delay_ms(300);
        Rev();
        delay_ms(200);
        HR();
        delay_ms(500);
        gr = 1;
        getgreen = 10;
    }

    else if ((getsonarR>380)&&(getsonarL<378))
    {
        HL();
        delay_ms(300);
        Stop();
    }
    else if ((getsonarR<380)&&(getsonarL>378))
    {
        MR();
        delay_ms(100);
        Stop();
    }
}
}
}

```