University of Florida

Department of Electrical and Computer Engineering

EEL5666

Intelligent Machine Design Lab

Spring 2006

Drew Lucas

4/25/06

Final Written Report

Table of Contents

Abstract

Many robots have been created that can traverse a flat surfaces in a controlled environment. My robot will be able to negotiate stairs while following a line. This will be achieved by using a novel form of locomotion which will allow the robot to reposition its center of mass so that it can lift itself up stairs. It will accomplish this by using various motors, sensors and systems to interpret its surroundings following predefined methods of movement.

Executive Summary

STAN (stair traversing autonomous navigator) is a robot designed to climb stairs. Most robots designed in IMDL simply roll around on the floor in two dimensions. I decided to make a robot that can move in three dimensions and climb a stair. This is achieved by rotating two lifting arms, one with a driving platform attached to it and the other with a counterweight attached. After the robot finds a stair, the platform arm swings onto the top of the stair and the counterweight turns and starts to shift the center of mass of the robot onto the stair. When the center of mass is on the stair, the platform motor is turned on causing the main platform to invert on to the stair. The robot then drives onto the stair and repositions itself to its original orientation so that it can continue climbing stairs. These procedures are completed by using a suite of various sensors that tell the microcontroller how to proceed. The project turned out to be harder to complete than originally thought. Numerous hardware issues arose and several times the lifting arms broke. I am pleased with the final robot but at the same time aware that there are several improvements that can be made in the future. I learned a great deal about sensors, programming the AVR microcontroller and electronics in general.

Introduction

This report will serve as an in depth description of how the robot performed and how it was created. The robot contains IR proximity sensors, bump detection, an IR line detector, and magnetic trip sensors that will allow for contactless switching. The robot is made out of plywood balsa milled out using the T-Tech CNC machine in lab. Using AutoCAD to test design ideas allowed for testing of positions without wasting wood. The object of the robot is to climb stairs and report how many stairs it has climbed and follow the line to the next stair.

Integrated Systems

The robot will follow a line by detecting a high contrast line with an IR sensor. While following the line, the robot continuously looks for a stair to climb. The robot reports back information through an LCD display. If it is, the robot will find the line again and climb the stair. The orientation sensor will be used during the lifting process to speed up or slow down the lifting so the robot will not flip over. Once the robot gets up onto the step, it will find the line again and continue to follow it. The robot also has "end of the world" detection so that if it is at the top step, it will report how many steps it has gone up. The platform is specially made for the lifting operation. All electronics, motors and power supplies are affixed on the platform in strategic locations so that during lifting and repositioning, the center of mass of the robot is in an acceptable place. The robot is controlled by an Atmel Atmega128 board from bdmicro.com. This processor was more than adequate for my robot's operation and because it is based on AVR, the web community allows you to get support if needed. Actually, I do not recommend this board unless you really need all the special functions listed on their website. Instead, I suggest buying a much cheaper board at sparkfun.com based on the Atmega128 which simply offers pin-outs for the chip. Programming was done in the C language using avrgcc compiler in Programmers Notepad. Downloading onto the board was done with Ponyprog and had generally trouble free operation. Interrupts are not used in this

instance accept for proximity sensing and timing because the clock speed is so fast, necessary operations can be carried out using simple polling. An LCD screen is used for debugging purposes and to display how many stairs the robot climbs. Power for the robot comes from a pack of 8 rechargeable NiMH batteries each rated at 2800mAH and 1.5V. This power supplies the six motors used during operation. Power for the board is provided by a 9v battery plugged into the 5v regulator on the controller board. There are 4 motor driver board with 2 drivers on each, 3 boards being used at one time with a spare in case a chip blows. The boards were made in lab using a design created by William Dubel.

Mobile Platform

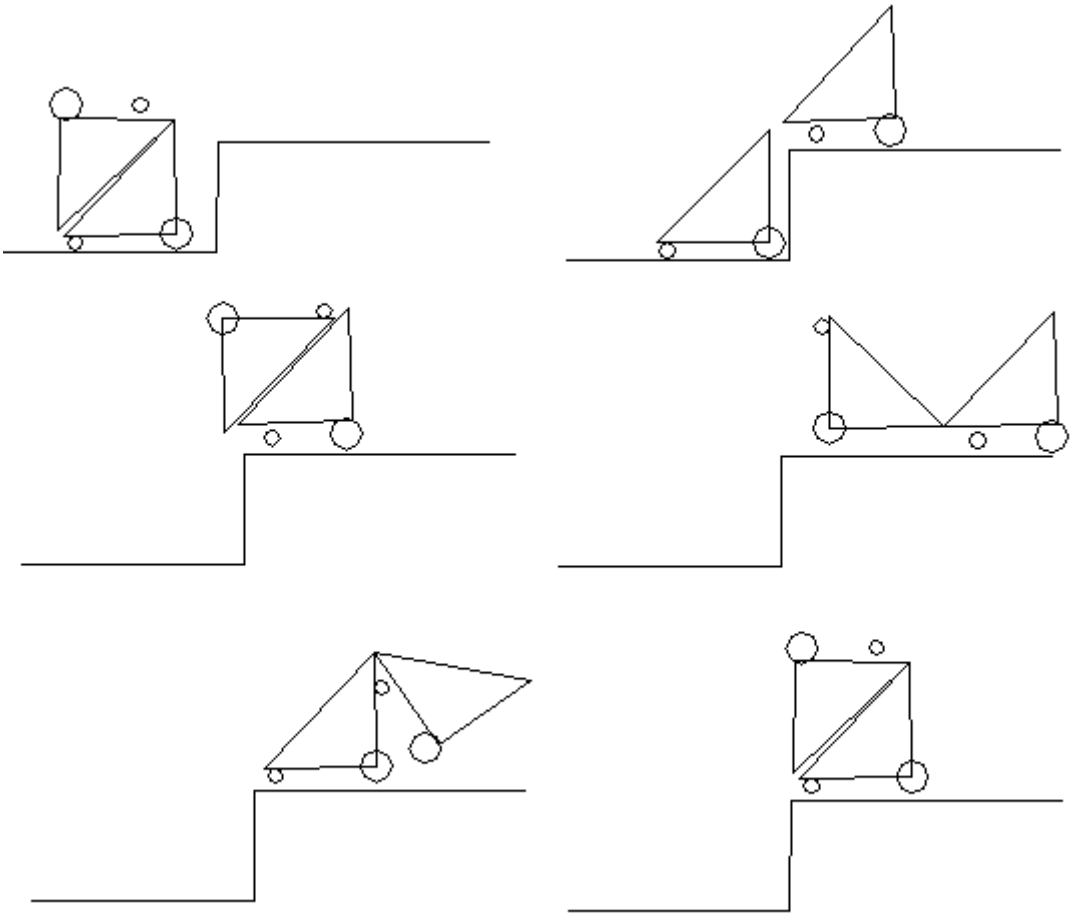
The mobile platform was constructed with 1/8" thick balsa plywood provided in lab. This was the most efficient way to make a customized shape because the design was first made on AutoCAD and moved around until an optimal positioning was found. Mounting holes were drilled after the sections were milled out and glued together so the placement of sensors, motors and electronics would be accurate. The basic organization of the robot starts with two motors at the bottom of the platform and a castor ball bearing wheel. The main chassis of the robot is oblong with the two lifting arms on the middle of the platform. One arm has a counterweight and the other a driving platform. The sensors

are mounted at strategic points on the robots to allow for optimal readings. The front and back of the robot is balanced to allow for stable lifting. Structural considerations in the design of the robot are very important because of the relatively large torques that the robot experiences while going up the stairs. Extra cross bracing were included in the design to account for the shearing caused by the torque from the lifting arms. Also, extra reinforcement was placed around the main lifting motors because of their own weight and all of the torque that they have to deal with. Originally, both arms were going to be positioned in the retracted position while moving around following the line. Original calculations allowed for this but after everything was assembled, the robot could only drive when the top platform counteracted the counterweight arm. Also, the wheel hubs connected to the lifting arms were a source of trouble. Two different times the hub fell off the aluminum arm and had to be reattached with JB weld. After the second time, a TIG welder was used to permanently attach the hubs to the arm.

Actuation

The robot moves around with the two motors and a castor wheel on the bottom of the platform which allows it to turn 360 degrees. Originally hacked servos were used on the bottom of the platform but it was quickly found that servos were not powerful enough to move the platform around because of the shear weight of the design. The servos were

rated at 57 oz.-in of torque whereas the motors were rated at 123 oz.-in of torque at 13rpm. The wheels were chosen because they have a tread on them so that they can grip the floor better. The lifting is done with two motors rated at around 300kg-cm which turns at 13 rpm. These motors were more than adequate to do the lifting movements with the weights involved. The low rpm also prevented the weights from being thrown around at high speeds. Motor movement is done by providing a PWM signal from the microcontroller to the driver boards which translates that signal into an output voltage. As per Adam's suggestion in class, the PWM signal actually is fed into the direction pin on the motor driver so that one variable in the code can change both direction and speed while avoiding current spikes.



Sensors

The sensors that are incorporated into the robot are an IR contrast detector, proximity sensors, bump sensors, and magnetic trip sensors that allow for a switching without contact. Originally a tilt sensor was going to be used to dynamically adjust the speed of the lifting arm so that it doesn't swing over too fast. However, after the platform was assembled, it was found that this sensor was superfluous considering the low max rpm of the lifting motors.

IR contrast detector: Ordered from lynxmotion.com. This sensor allows the robot to dynamically adjust its motion depending on where a high contrast line is. Originally I thought that I could easily reproduce line following by just detecting the change in contrast. However, a complex algorithm had to be formulated to stay within the line while driving. If ever doing line following again, I would definitely buy a 2 or 3 sensor line following board. These boards are not much more expensive and can provide accurate simplified line following.

Proximity sensors: Ordered from sparkfun.com. These sensors are used to detect the stair and "edge of the world" situations. They are rated for accurate distance measurements from 10 to 80cm. The sensors are calibrated at the beginning of operation so that the specific value outputs are not important. A simple algorithm is used to detect if there is a change from the calibrated value and this is used to detect if there is a stair or edge.

Bump sensors: Ordered from mouser.com. The bump switches send signals to tell the processor to respond when the arms are in certain positions and when the robot hits walls. The switches chosen for the robot were the lowest operation force that could be found so that a very small force will trip the switch.

Magnetic Trip: These sensors are placed at specific positions to detect when the robot gets to different orientations. The fact that these sensors trip without contact is an important facet because it the high torque motors are kept on for longer than necessary, the robot frame could be destroyed.

Behaviors

The robot demonstrates autonomous stair climbing. This behavior is not in itself a complicated programming task. No complex decision making was necessary for the microcontroller. The hard part about this robot was balancing all of the forces involved during lifting. This, along with getting all of the driver boards to function correctly, took up most of the design time. Line following was not as much of a success as was desired. My plan for using a single IR detector for detecting change in contrast was good in theory but the errors in after each line detection compounded on top of each other leading to very sloppy line following. Since line following was not the main aim of this robot, its completion was left until the end of the project. Consequently, there was not enough time left to order a triple IR line detector that would greatly facilitate line tracking.

Another problem encountered was that since the wheels were so far apart, turning the platform required a relatively large radius.

Conclusion

Before taking this class I read many times on the class website just how much time that the class takes. I didn't really take this to heart just as someone reading this might not. This class was, however, the best class that I've ever taken in college. I've learned so much about not only control theory and electronics but also about how to budget my time appropriately. Even though this project has not completed all of the objectives outlined at the beginning of the semester, it has achieved its main task of climbing stairs.

I've learned some very important, hard earned lessons this semester. First, I found out very quickly to double check every electrical connection I make. I probably went through 3-4 motor driver boards because I wasn't paying attention to what I was doing. I also wasted many long hours troubleshooting code when in reality it was a simple hardware connection. Also, I used wire-wrap extensively in my platform after crimpling wires lead to faulty connections.

In the future, I look forward to refining my design and code to get the other behaviors that I planned for the robot. Those include checking if there is an obstacle on the stair before it climbs it and smooth line tracking.

Special thanks to Adam Barnett and Sara Keen for all their help in lab and to Etan Shaul for his help with programming.

References

Atmel ATMega128 Manual

http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

LCD Manuals

http://www.doc.ic.ac.uk/~ih/doc/lcd/

AVR Documentation

http://www.nongnu.org/avr-libc/user-manual/

Appendices

C code

```
#include <avr/io.h>
#include <avr/interrupt.h>

#include <string.h>
#include <ctype.h>
#include <stdio.h>
#include <inttypes.h>
#include <math.h>

#define rd   OCR3A
#define ld   OCR3B
#define rm   OCR3C
#define rp   OCR1A
#define lp   OCR1B
#define lm   OCR1C

#define line_follow 0x01
#define bump_left 0x02
#define bump_right 0x04
#define bump_plat 0x08
#define mag_body 0x10
#define mag_arm 0x20



//////////////////////COUNT Start (from
bdmicro.com)////////////////////////////////////////////////
volatile uint16_t ms_count;
void sleep(void)
{
  volatile uint16_t count;
  count=0;
  for(count=0;count<=2000;count++);

}

void ms_sleep(uint16_t ms)
{
  TCNT0  = 0;
  ms_count = 0;
  while (ms_count != ms)
    ;
}

/*
 * millisecond counter interrupt vector
 */
SIGNAL(SIG_OUTPUT_COMPARE0)
```

```
{
  ms_count++;
}

/*
 * initialize timer 0 to generate an interrupt every millisecond.
 */
void init_timer(void)
{
  /*
   * Initialize timer0 to generate an output compare interrupt, and
   * set the output compare register so that we get that interrupt
   * every millisecond.
   */
  TIFR  |= _BV(OCIE0);
  TCCR0  = _BV(WGM01)|_BV(CS02)|_BV(CS00); /* CTC, prescale = 128 */
  TCNT0  = 0;
  TIMSK |= _BV(OCIE0);    /* enable output compare interrupt */
  OCR0   = 125;          /* match in 1 ms */
}

////////////////////////COUNT
END////////////////////////////////////////////////////

////////////////////////LCD
Start////////////////////////////////////////////////////
void init_lcd(void)
{
      DDRA=0xFF;

      PORTA=0x03;
      sleep();
      PORTA=0X13;
      sleep();
      PORTA=0x03;
      sleep();

      PORTA=0x03;
      sleep();
      PORTA=0X13;
      sleep();
      PORTA=0x03;
      sleep();

      PORTA=0x03;
      sleep();
      PORTA=0X13;
      sleep();
      PORTA=0x03;
      sleep();

      PORTA=0x02;  /*4bit mode*/
      sleep();
      PORTA=0x12;
      sleep();
      PORTA=0x02;
      sleep();
```

```
PORTA=0x02; /*interface length*/
sleep();
PORTA=0x12;
sleep();
PORTA=0x02;
sleep();
PORTA=0x08;
sleep();
PORTA=0x18;
sleep();
PORTA=0x08;
sleep();

PORTA=0x00; /*off*/
sleep();
PORTA=0x10;
sleep();
PORTA=0x00;
sleep();
PORTA=0x08;
sleep();
PORTA=0x18;
sleep();
PORTA=0x08;
sleep();

PORTA=0x00; /*clear*/
sleep();
PORTA=0x10;
sleep();
PORTA=0x00;
sleep();
PORTA=0x01;
sleep();
PORTA=0x11;
sleep();
PORTA=0x01;
sleep();

PORTA=0x00; /*Cursor move direction*/
sleep();
PORTA=0x10;
sleep();
PORTA=0x00;
sleep();
PORTA=0x06;
sleep();
PORTA=0x16;
sleep();
PORTA=0x06;
sleep();

PORTA=0x00; /*Enable display*/
sleep();
PORTA=0x10;
sleep();
```

```
	PORTA=0x00;
	sleep();
	PORTA=0x0F;
	sleep();
	PORTA=0x1F;
	sleep();
	PORTA=0x0F;
	sleep();

}

void write_char_lcd(char sample)
{
	int tog1;
	int tog2;
	int nib1;
	int nib2;
	int temp;
	int getnib1;

	getnib1 = 0x0F;
	tog1 = 0x20;
	tog2 = 0x30;
	sample = (unsigned int)sample;

	nib1 = sample >> 4;

	temp = nib1 | tog1;
	PORTA=temp;
	sleep();
	temp = nib1 | tog2;
	PORTA=temp;
	sleep();
	temp = nib1 | tog1;
	PORTA=temp;
	sleep();

	nib2= sample & getnib1;

	temp = nib2 | tog1;
	PORTA=temp;
	sleep();
	temp = nib2 | tog2;
	PORTA=temp;
	sleep();
	temp = nib2 | tog1;
	PORTA=temp;
	sleep();

}
void write_num_lcd(uint16_t sample)
{
	int tog1;
	int tog2;
	int nib1;
	int nib2;
	int temp;
```

```
	int getnib1;

	getnib1 = 0x0F;
	tog1 = 0x20;
	tog2 = 0x30;


	nib1 = sample >> 4;

	temp = nib1 | tog1;
	PORTA=temp;
	sleep();
	temp = nib1 | tog2;
	PORTA=temp;
	sleep();
	temp = nib1 | tog1;
	PORTA=temp;
	sleep();

	nib2= sample & getnib1;

	temp = nib2 | tog1;
	PORTA=temp;
	sleep();
	temp = nib2 | tog2;
	PORTA=temp;
	sleep();
	temp = nib2 | tog1;
	PORTA=temp;
	sleep();

}

void write_string_lcd(char string[] )
{	int size;
	int i;
	size=strlen(string);

	for(i=0;i<size;i++)
		{write_char_lcd(string[i]);}
}
void lcd_clear(void)
{
	PORTA=0x00; /*clear*/
	sleep();
	PORTA=0x10;
	sleep();
	PORTA=0x00;
	sleep();
	PORTA=0x01;
	sleep();
	PORTA=0x11;
	sleep();
	PORTA=0x01;
	sleep();
}
```

```
//////////////////////LCD
END//////////////////////////////////////////////////////

//////////////////////IO Start
////////////////////////////////////////////////
void init_input(void)
{
      DDRC=0x00;
      PORTC=0XFF;
      int line;
      line=PINC;

      DDRE=0xFF;
      DDRB=0xFF;

      TCCR1A = 0xA9;
      TCCR1B = 0x0A;
      TCCR3A = 0xA9;
      TCCR3B = 0x0A;
}
//////////////////////IO
END//////////////////////////////////////////////////////

//////////////////////Lifting Operations
Start////////////////////////////////////
void drive_to_wall(void)
{     ld=180;
      rd=180;
      write_string_lcd("wall found waiting for bumps");

      while(!(((PINC & 0x02)==0x00 && (PINC & 0x04)==0x00)))
            {if((PINC & 0x02)==0x00){ld=128;}//leftmotor off
             if((PINC & 0x04)==0x00){rd=128;}//rightmotor off
            }
      lcd_clear();
      write_string_lcd("bumps triggered");
      ms_sleep(1500);
      ld=128;
      rd=128;
}
void platform_stage(void)
{
      while(!((PINC & 0x08)==0x00))
            {
             if(!((PINC &
0x08)==0x00)){rm=85;write_string_lcd("stageing platform");lcd_clear();}
             }
      rm=128;write_string_lcd("platform bumped");
      ms_sleep(1500);
      lcd_clear();
}

void weight_stage(void)
{
      while(!(PINC & 0x20)==0x00)
            {write_string_lcd("weight
stage");lcd_clear();lm=50;rm=128;}
```

```
      lm=128;

}

void weight_lift(void)
{
      if((PINC & 0x20)==0x00){lm=160;}
      else{lm=128;}
}

void lift_body_stair(void)
{
      weight_stage();
      while(!((PINC & 0x10)==0x00))
            {write_string_lcd("lift body");lcd_clear();

             rm=170;
             weight_lift();}
      lm=70;
      rm=128;
      ms_sleep(2000);
}

void check_stair(void);

//turn 90deg
//drive backward/forward while reading value at 10ms increment
//variable if change is greater than value

//if ok -> turn 90 degrees when line detect and drive to wall-> exit
function
//else detect obsticle ->go back to line turn 90 degrees the other way
//and line follow and obsticale avoid while(1)


void on_stair_drive(void)
{write_string_lcd("on stair
drive");rd=128;ld=128;lm=70;rm=128;lp=185;rp=185;ms_sleep(4000);lp=128;
rp=128;lm=70;rm=128;}

void on_stair_reposition(void)
{     lcd_clear();
      write_string_lcd("reposition");
      lm=70;
      rm=128;
      rm=60;
      lm=50;
      ms_sleep(2000);
      lm=70;
      ms_sleep(15500);
      rm=128;
      ms_sleep(2000);
      while(!((PINC & 0x10)==0x00))
      {lm=140;rm=128;}
      lm=70;
      rm=180;
      ms_sleep(3000);
```

```
      rm=128;
      //find line and repeat big loop
      //increment stair counter
}



//////////////////////Lifting Operations
END//////////////////////////////////////

//////////////////////AD Conversion Start(from
bdmicro.com)//////////////////////////////////////

/*
 * adc_init() - initialize A/D converter
 *
 * Initialize A/D converter to free running, start conversion, use
 * internal 5.0V reference, pre-scale ADC clock to 125 kHz (assuming
 * 16 MHz MCU clock)
 */
void adc_init(void)
{
  /* configure ADC port (PORTF) as input */
  DDRF  = 0x00;
  PORTF = 0x00;

  ADMUX = _BV(REFS0);
  ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR) |
_BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0);
}

/*
 * adc_chsel() - A/D Channel Select
 *
 * Select the specified A/D channel for the next conversion
 */
void adc_chsel(uint8_t channel)
{
  /* select channel */
  ADMUX = (ADMUX & 0xe0) | (channel & 0x07);
}


/*
 * adc_wait() - A/D Wait for conversion
 *
 * Wait for conversion complete.
 */
void adc_wait(void)
{
  /* wait for last conversion to complete */
  while ((ADCSR & _BV(ADIF)) == 0)
    ;
}

/*
 * adc_start() - A/D start conversion
```

```
 *
 * Start an A/D conversion on the selected channel
 */
void adc_start(void)
{
  /* clear conversion, start another conversion */
  ADCSR |= _BV(ADIF);
}

/*
 * adc_read() - A/D Converter - read channel
 *
 * Read the currently selected A/D Converter channel.
 */
uint16_t adc_read(void)
{
  return ADC;
}

/*
 * adc_readn() - A/D Converter, read multiple times and average
 *
 * Read the specified A/D channel 'n' times and return the average of
 * the samples
 */
uint16_t adc_readn(uint8_t channel, uint8_t n)
{
  uint16_t t;
  uint8_t i;

  adc_chsel(channel);
  adc_start();
  adc_wait();

  adc_start();

  /* sample selected channel n times, take the average */
  t = 0;
  for (i=0; i<n; i++) {
    adc_wait();
    t += adc_read();
    adc_start();
  }

  /* return the average of n samples */
  return t / n;
}

////////////////////////AD Conversion
END/////////////////////////////////////////////

////////////////////////Motor Functions
Start/////////////////////////////////////////
void motors(int left_desired_speed , int right_desired_speed,int scale)
{
      while( (ld != left_desired_speed) || (rd != right_desired_speed)
)
```

```
 {
      if (left_desired_speed > ld)
      {ld = ld + scale;ms_sleep(10);}
      else if( left_desired_speed < ld )
      {ld = ld - scale;ms_sleep(10);}
      if( right_desired_speed > rd )
      {rd = rd + scale;ms_sleep(10);}
      else if( right_desired_speed < rd )
      {rd = rd - scale;ms_sleep(10);}
 }

}
//////////////////////Motor Functions
End//////////////////////////////////////

int main(void)
{
      sei();
      init_timer();
      init_lcd();

      adc_init();
      init_input();

      int ground;
      int stair_number=0;
      lm=70;
      rm=128;
      rd=180;
      ld=180;
      lp=128;
      rp=128;


lcd_clear();write_string_lcd("ON");
ms_sleep(2000);


ground=adc_readn(0, 20);
int line=2;


ld=180;rd=180;
while((PINC & 0x01)==0x01){;}



while(1)
{

      ms_sleep(50);

      if((PINC & 0x01)==0x00){rd=180;ld=180;}

      if((PINC & 0x01)==0x01)
      {
```

```
            if(line==1)
                  {line+1;

                   ld=190;rd=80;
                   while((PINC & 0x01)==0x01){;}
                  ld=180;rd=180;
                  }
            if(line==2)
                  {line-1;

                   ld=80;rd=190;
                   while((PINC & 0x01)==0x01){;}
                   ld=180;rd=180;
                  }

      }


if(adc_readn(0, 20)>(ground+25)){ld=130;}
if(adc_readn(0, 20)>(ground+25)){rd=130;}

if((adc_readn(0, 20)>(ground+15)) && (adc_readn(1, 20)>(ground+15)))
{drive_to_wall();
platform_stage();
lift_body_stair();
on_stair_drive();
on_stair_reposition();
lcd_clear();write_string_lcd("End Main");
stair_number=+1;
line=2;
ground=adc_readn(0, 20);
lcd_clear();write_string_lcd(stair_number);write_string_lcd(" Stairs
Climbed");ms_sleep(3000);
line=2;ld=180;rd=180;
while((PINC & 0x01)==0x01){;}}
}

}

}
```