Johnny-5

Life Imitator

Matt Raines
EEL 5666 -- Intelligent Machines Design Laboratory
Summer 2002

Professors: Dr. A. Antonio Arroyo and Dr. Eric M. Schwartz
Assistants: Tae Choi and Uriel Rodriguez

Table of Contents

Abstract

Johnny-5 is a mobile autonomous agent that imitates a living creature. The robot roams his territory performing a variety of behaviors chosen based on its mood. Johnny-5 uses bump and IR sensors to detect obstacles in its path, CdS photocells to detect light and dark areas, and a voltage divider circuit to monitor battery levels. Behaviors include object avoidance, light following, light avoiding, hunger signaling, wandering, dancing, and vocalizing mood. Johhny-5's mood ranges from angry to happy and is determined by factors such as battery level, number of avoidances, number of collisions, and ambient light.

Executive Summary

Johnny-5 was designed to imitate a living creature. Sensors and behaviors have been integrated to allow it to react to its environment in a manner similar to a household pet such as a dog.

Johnny-5 uses four types of sensors to interpret its environment. Three bump sensors detect collisions while three infrared emitters and two detectors warn of nearby objects. A voltage divider circuit connected to the batteries and an analog input allows it to constantly monitor its energy level. Finally, two cadmium-sulfide photoresistors detect the amount of light present.

Johnny-5 behaves in different ways based on its current mood. Each mood is a behavior in itself, comprised of several lower-level behaviors shared by each mood. There are four moods ranging from angry to happy with two intermediates. They are triggered by both external and internal factors including number of collisions or obstacles to avoid, level of lighting in its environment, previous mood, and energy level. The behaviors that require direct input from sensors are obstacle avoidance, light following, light avoiding and hunger signaling. Johnny-5 is also capable of aimless wandering and performing the Dance of the Robot, as well as communicating its mood to the surrounding environment using visible-light LEDs and a piezo buzzer.

. Johnny-5 is controlled by the MC68HC11 microcontroller with 32 KB memory, powered by six rechargeable NiCd AA batteries, and actuated with two modified servomotors.

Introduction

Observed in the proper perspective, animals can be seen as complex systems that respond to their environment in predictable ways. This project is an experiment in the simulation of animal behavior. Johnny-5 senses its environment using bump, infrared, voltage, and light sensors. This data is processed in a microcontroller serving as the robot brain, which instructs the robot body to behave in predictable ways. Due to the incredible complexity of living creatures and the limited experience of the designer, behavior concessions had to be made and thus the programmed behaviors are simplified versions of the way a biological animal reacts to its environment. These behaviors include collision avoidance, following and avoiding light sources, energy seeking, wandering, and dancing. Behavior is implemented with the idea that unhealthy situations anger an animal, while healthy situations please it. Johnny-5 decides how to behave based on its mood, which is determined by evaluations of both its physical state and its surrounding environment. As with nature, Johnny-5 will never be a finished product but will serve as a stepping-stone for further experimentation with models of animal behavior.

Integrated System

Johnny-5 consists of a microprocessor, servomotors, wheels, batteries, and various sensors all mounted on a platform similar to the body of the Mekatronix TJPro. Its brain is the Mekatronix MTJPRO11 microcontroller, a Motorola MC68HC11 microcontroller with 32 KB of SRAM, eight analog inputs, four digital inputs, and eight digital outputs. The MTJPRO11 is mounted inside the body for protection. This robot brain reads input from sensors and provides storage for the programs that tell Johnny-5 what to do with the information it receives. The robot acts based on sensor data and software algorithms. Servomotors and wheels mounted on each side are used for movement of the robot. The robot uses three bump sensors arrayed around its front to determine if it has run into an object. Infrared emitters and detectors mounted at the front of the platform inform if it is nearing objects so that it may maneuver away. The robot will monitor its power level using a voltage divider circuit connected to its batteries and an analog input. Two light detectors differentiate between lighter and darker spaces and are directed 90 degrees apart to allow it to determine the direction of the light source. All these are mounted on a lightweight wooden platform.

Mobile Platform

The body of the robot is constructed out of 1/4-inch aircraft plywood. The wood was cut by a T-tech machine available in lab and using a plot generated using AutoCAD software. The top platform is circular to allow for better obstacle avoidance. The inside of the platform has space to store the microcontroller, motors, and batteries. Johnny-5's head is attached to a smaller platform raised above the circular one and he wears a cape to hide his nakedness. Johnny-5 is approximately 10 inches tall, 6 inches long and 7 inches wide.

Actuation

Two 3-inch wheels on opposite sides of the platform provide traction for the robot to move and two smooth contact points in the rear reduce friction and improve stability.

Each wheel is rotated by its own servomotor. The motors are modified Diamond Precision Standard Servos obtained from Mekatronix and are hacked to allow a constant360 degrees of motion. Despite the name these motors are not high-precision machines, and some differences in output torque are exhibited. An array of batteries is located within the body of the robot to provide operating power.

Sensors

Bump Sensors

Purchased from: Mekatronix
316 NW 17 th St., Suite A
Gainesville, FL 32603
Phone: 352-376-7373
www.mekatronix.com

Three bump sensors are spaced around the front of the platform. One is oriented forward and the other two approximately 35 degrees to either side. Thin wooden bumpers attached ahead of the switches insure that any front collision will be detected. The switches are connected to digital inputs of the microcontroller. The software for these sensors can be found in the appendix.

Infrared LEDs

Purchased from: Radio Shack
2 NW 16th Ave
Gainesville, FL 32601
(352) 375-2336
www.radioshack.com

Three infrared LEDs are mounted atop the front of the platform. The LEDs are glued inside of barrels constructed out of ink pen caps to provide a focused beam of light.

One is directed straight ahead and the other two are directed approximately 35 degrees to either side. This orientation ensures adequate coverage of the area in front of the robot. The infrared emitters are connected to three of the digital outputs on the MTJPRO11.

Sharp Infrared Detectors

Two Sharp IR detectors obtained from lab materials from previous classes act as short-range eyes for Johnny-5. These are mounted at the front on the underside of the platform. This allows the sensors to detect infrared light reflected off of objects in front of the robot while shielding them from ambient sources. These IR detectors had to be hacked to allow an analog signal instead of a digital one. The procedure for hacking these can be found on the IMDL website: IR Hack. The analog signal allows each detector to be connected to its own analog port, allowing comparison of the two values detected. The two IR detectors have different base readings and thus signal different values for objects at the same distance. Testing with various colors and textures of objects revealed the best correction—a sort of digital monocle for the right side detector. Another problem encountered with the right side detector was an extreme sensitivity to ambient light that did not occur with the left side detector. The solution eventually arrived at was to only partially replace the cover on the right side detector. This prevented erroneous signaling when ambient light was detected, though the mechanism by which this worked is unknown. The effective distance for object detection is around 18 inches for white or other light-colored objects and around 8 inches for black and other dark objects. The software algorithm that integrates the infrared emitters and detectors and allows these sensors to help Johnny-5 avoid collisions can be found in the appendix.

Battery Level Circuit

The energy level sensor is a simple design constructed from wire, headers, and a single resistor. The sensor is mounted inside the robot platform because it monitors internal power levels and need not interact with the Johnny-5's environment. The circuit is a voltage divider that delivers a fraction of the total voltage remaining in the batteries to the signal input of its analog port. The sensor gives a maximum reading of 255 until the voltage falls below 7.45V, at 4.68V the signal is 188 and the robot no longer has enough power to operate. The main difficulty in constructing this sensor was selecting the right resistor value to properly scale the voltage delivered to the input. Eventually, it was discovered that a 1 MΩ resistor is the best choice. The accompanying circuit diagram details the voltage divider circuit and the table lists the various signal values and their respective voltages. The voltage remaining can be obtained by simply reading the analog port the sensor is connected to.

Cadmium Sulfide Photoresistors

Purchased from:	Electronics Plus 2026 SW 34th Street Gainesville, FL 352-371-3223

Two light-detecting photoresistors mounted on either side of the platform detect the amount of light present. They are mounted at a forward angle to provide the best range of detection for their light-following purpose. The Figure 3 shows their orientation and fields of detection. From the diagram it is apparent that Johnny-5 cannot detect light sources behind it. This imperfection is consistent with biology, however, as most animals do not have eyes in the back of their head. These sensors use a voltage divider circuit to detect the intensity of the light. The photoresistors themselves change their electrical resistance by the amount of light hitting them and thus the fraction of a total voltage dissipated by them indicates the lighting level. The lower the light level, the higher the signal value. Experimentation revealed that the other resistance in the voltage divider circuit should be 48 kΩ to best fit the application. This resistance was obtained with the series combination of a 33kΩ and a 15kΩ resistor. Figure 4 shows the voltage divider circuit used. The software for determining the direction of the light source is given in the appendix.

Behaviors

Obstacle Avoidance

The most basic behavior exhibited by Johnny-5 is object avoidance. This is accomplished through the integrated use of bump sensors, IR emitters, and IR detectors. When the IR detectors indicate an object ahead, the robot turns to avoid it. If the object is straight ahead or primarily on the right side, the robot turns left until there is no longer an obstruction ahead. If the object is on the left side, it turns right until its path is clear again. Should the IR system fail and Johnny-5 collide with an object, the bump routine cycles. When a bump sensor indicates there has been a collision, the robot backs up, then turns randomly and heads in a new direction if the way is clear. Johnny-5 will avoid obstacles up to 1 foot away, though various physical properties of the obstruction may cause the IR to detect it at a shorter distance. Originally, there were only two IR emitters mounted at an angle on the platform. This left a blind spot directly ahead which caused a problem. Mounting an additional IR emitter directed straight ahead solved this. The avoidance code can be found in the appendix.

Energy Level Monitoring

Another basic behavior is monitoring the power level remaining in the batteries. The battery level circuit is used to detect the voltage left. The scaled voltage delivered to the analog port is converted to a hexadecimal value and Johnny-5 uses this value to determine if he is low on energy. A reading of 255 indicates full or near-full batteries and a reading of 188 indicates the robot is moments away from dying. When it detects a battery level of 190, Johnny-5 will spin and buzz pleadingly to be fed. Difficulty was encountered in determining the voltage level required for the robot to operate. Only batteries were available at home to test different voltages. Using the variable voltage sources available in the lab solved this problem. The energy-level code can be found in the appendix.

Light Following

Johnny-5 uses its photoresistor sensors to detect and follow light sources. By turning towards the side with the higher light intensity (lower signal value) until both sensors read the same value, the robot can track a moving light source. If the robot turns too far, the sensors will detect it and rotate it the opposite direction until the light is once again centered. Once the source is centered ahead of it, Johnny-5 will move towards the source until it moves or the behavior times out. The sensors are not precise components, so adjustments had to be made in order for the robot to properly center itself on the light source. The right sensor consistently reads higher than the left for the same levels of intensity. Adjusting the sensor value in the software solved this problem. The light following behavior algorithm can be found in the appendix

Light Avoiding

Johnny-5's light avoiding behavior also uses the photoresistor sensors. The light source or area of most intense light is detected the same way as for the light following behavior. Once the robot is centered on the source, it will turn 180 degrees and move away. It will stop occasionally to reorient itself away from the light. The light avoiding code can be found in the appendix.

Wandering

By generating a (semi)random number and using it to time turns, Johnny-5 will turn in a random direction. The random number is generated using the TCNT function of the HC11 and the mathematical mod function to scale it to a usable number and is based on code written by Ivan Zapata and provided on the IMDL website: 'random' example. Johnny-5 will turn for a time based on the number returned by the random function, then move forward until a new number is generated.

Dancing

By executing a programmed series of turns and forward and backward movements, Johnny-5 will perform the Dance of the Robot.

Moods

Johnny-5's moods are determined by both internal and external factors. Mood is continually updated in software and each mood uses some or all of the behaviors to express itself. Each mood is given a whole number value between 1 and 5, 1 being happy and 5 being angry. The battery level either improves or worsens mood a variable amount depending on the value returned by the sensor. Every collision adds one to the mood variable, angering the robot. Every time Johnny-5 must avoid an obstacle, 0.2 is added to the mood, still bothering it but at a slower rate. Higher light intensities please the robot and decrease the mood variable by an amount proportional to the light intensity. Similarly, a darker environment increases the mood variable, angering Johnny-5.

Happy

If Johnny-5 is happy, he is feeling his best and has five behaviors to choose from. Which behavior is executed is decided randomly, but with more probability for some behaviors than for others. He might wander, seek light, dance, or sing one of two songs he knows, though he is more likely to sing his happiest song than any other behavior. Each behavior is timed, and once it times out, a new behavior is randomly selected.

Pleased

When Johnny-5 is pleased, he is not quite happy, but still feeling positive. He might seek out light, wander, dance, or sing. He can only sing one song while pleased, the other he always saves for when he is happiest. Also, he is more likely to seek light or wander than to sing and dance. Again, each behavior is timed and randomly selected.

Disgruntled

When Johnny-5 starts to get bothered by the things going on inside and around him, he gets disgruntled. A random selection of wandering, avoiding light, and buzzing angrily characterize this mood. He is far more likely to simply gripe about his dissatisfaction than to wander or bother with lights. When one behavior times out, a new is selected randomly.

Angry

When things get too bothersome for Johnny-5, he gets mad. He is more likely to buzz and avoid light than to wander or seek out light for retribution. Again, the selected behavior is timed and selected randomly.

Communicating Mood

To properly let others know how he is feeling, Johnny-5 must be able to communicate mood effectively. This is accomplished through visible light LEDs and a piezo buzzer. The LEDs are his eyes and flash in different ways when is in different moods. When in a positive mood, Johnny-5 winks to the world around him and blinks slowly and when he is negative, his eyes blink quickly and angrily. Johnny-5 sings his two songs and buzzes gripingly using the piezo buzzer. Higher tones indicate higher spirits and lower tones mean anger.

Conclusion

Overall, Johnny-5 was a success in that it meets the goals set at the beginning of the project, though in a slightly different manner and to a less precise degree than was originally planned. All behaviors are performed correctly, if not necessarily precisely. There are many and great limitations on the original goals of the project. Inexpensive equipment malfunctioned easily and either had to be replaced or made do with depending on time and money constraints. Expensive equipment was unattainable due to an independent undergraduate budget. The mood determining function is limited in scope of available moods as well as factors to determine those moods. This is due to the increased overall complexity added with each new mood and factor. In a less limited setting, more precise sensors and longer lasting servomotors would be implemented. Additional sensors such as motion detectors, microphones, and force sensors would also be integrated. A projectile device or mechanical arm would improve interaction with the environment as well as help communicate mood. A charging station would allow the robot to feed itself when hungry instead of depending on human assistance. Improvements can be made both in hardware and software to an infinite degree—evolution is never finished!

Documentation

All borrowed information can be found at IMDL Homepage

Ideas and Guidance: Dr. Arroyo, Dr. Schwartz, Tae Choi, Uriel Rodriguez

Keith L. Doty. TJPRO assembly manual; Mekatronix 1999.

Appendix

```
/*****************************************************/
/*****************************************************/
/******                   ***********************/
/*****    Combined Behaviors    **********************/
/*****  Matt Raines  8/8/2002  **********************/
/******                   ***********************/
/*****************************************************/
/*****************************************************/



/***** Includes *****/
#include <stdio.h>
#include <tjpbase.h>
#include <servotjp.h>
#include <analog.h>
#include <motortjp.h>
#include <vectors.h>



/***** Constants *****/
#define Lserv          2
#define Rserv          1
#define Lfwd           500
#define Rfwd           250
#define Lrev           250
#define Rrev           500
#define IRL_LMT                86
#define IRR_LMT        123
#define CHG_lmt        189
#define BUMPER         analog(0)
#define RGT_IR         analog(2)
#define LFT_IR         analog(5)
#define photoL         analog(3)
#define photoR         analog(1)
#define volts          analog(7)
#define lti            0x47
#define rti            0x87
#define buz            0x17
#define none           0x07
```

```
/***** Prototypes *****/

/*** Movement ***/
void forward(void);
void back(void);
void stop(void);
void turn_right(void);
void turn_left(void);
void turn_rand(void);
/*** Behaviors ***/
void avoid(void);
void to_light(void);
void from_light(void);
void power(void);
void wander(void);
void dance(void);
void five(void);
void two(void);
void one(void);
/*** States ***/
void angry(void);
void disgruntled(void);
void apathetic(void);
void pleased(void);
void happy(void);



/***** Global Variables *****/
int bumps, bumpprev, avoids, avoidprev;


/*****************************************************/
/*****************************************************/
/***********     ***************************************/
/********** MAIN *************************************/
/***********     ***************************************/
/*****************************************************/
/*****************************************************/



void main(void)
{
        int i,j, choice, collisions;
        int lightL, lightR, lightprev;
```

```
float mood, moodprev, avoidances, lighting, v;

init_analog();
init_motortjp();
init_clocktjp();
init_servotjp();

*(unsigned char *) 0x7000=0x07;

bumps = 0;
avoids = 0;
mood = 6;
moodprev = mood;
bumpprev = 0;
avoidprev = 0;
lightR = photoR;
lightprev = lightR;
while(1)
{

        power();
        v = volts - 188;
        if(v > 33)
                v = 1;
        else
                v = -0.5;

        collisions = bumps - bumpprev;
        avoidances = (avoids - avoidprev) * 0.2;

        lightR = photoR;
        if(lightR <= 175)
        {
                if((lightR + 2) < lightprev)
                        lighting = 175 / lightR;
                else
                        lighting = 0;
        }
        else
        {
                if((lightR - 2) > lightprev)
                        lighting = -255 / lightR;
                else
                        lighting = 0;
        }
        lightprev = lightR;
```

```
            mood = mood + v + collisions + avoidances + lighting;

            if(mood > moodprev)
                {
                    for(j=0; j < 4; j++)
                    {
                        for(i=0; i < 10000; i++)
                            *(unsigned char *) 0x7000=0xD7;
                        for(i=0; i < 10000; i++);
                        for(i=0; i < 10000; i++);
                        for(i=0; i < 10000; i++);
                        *(unsigned char *) 0x7000=0xC7;
                    }
                }
            if(mood < moodprev)
                for(j=0; j<500; j++)
                {
                    *(unsigned char *) 0x7000=(buz);
                    for(i=0; i<100; i++);
                    *(unsigned char *) 0x7000=(none);
                    for(i=0; i<100; i++);
                }

            if(mood < 2.5)
                angry();
            if((mood >= 2.5) & (mood < 5))
                disgruntled();
            if((mood >= 5) & (mood < 7.5))
                pleased();
            if(mood >= 7.5)
                happy();

            moodprev = mood;

        }

    }




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

```
/**********                **************************/
/********* MOVEMENT ROUTINES ************************/
/**********                **************************/
/****************************************************/
/****************************************************/



/***** FORWARD:  move forward *****/
void forward(void)
{
        servo(Rserv, Rfwd);
        servo(Lserv, Lfwd);
}



/***** BACK:  back up *****/
void back(void)
{
        servo(Rserv, Rrev);
        servo(Lserv, Lrev);
}



/***** STOP:  stop *****/
void stop(void)
{
        servo(Rserv, 0);
        servo(Lserv, 0);
}



/***** TURN_RIGHT:  rotates clockwise indefinitely *****/
void turn_right(void)
{
        servo(Rserv, Rrev);
        servo(Lserv, Lfwd);
}



/***** TURN_LEFT:  rotates counterclockwise indefinitely *****/
void turn_left(void)
```

```
{
        servo(Rserv, Rfwd);
        servo(Lserv, Lrev);
}



/***** TURN_RAND: turns randomly *****/
void turn_rand(void)
{
        int i;
        unsigned rand;

        rand = TCNT;

        if (rand & 0x0001)
                turn_right();
        else
                turn_left();

        /*** spin ***/
        i = rand % 2048;
        if(i > 1024)
                wait(i);
        else
                wait(1024);
}




/*****************************************************/
/*****************************************************/
/*****               *****************************/
/***** BEHAVIOR ROUTINES *****************************/
/*****               *****************************/
/*****************************************************/
/*****************************************************/



/***** AVOID:  avoid obstacles *****/
void avoid(void)
{
	int irdr, irdl;

	irdr = RGT_IR;
	irdl = LFT_IR;
	bumpprev = bumps;
	avoidprev = avoids;

	if(FRONT_BUMP)
	{

		bumps = bumpprev + 1;
		back();
		wait(1200);
		turn_rand();
	}

	if((irdr > IRR_LMT) || (irdl > IRL_LMT))
	{

		avoids = avoidprev + 1;

		while((irdr > IRR_LMT) || (irdl > IRL_LMT))
		{
			if((irdr-45) > irdl)
				turn_left();
			else
				turn_right();

			irdr = RGT_IR;
			irdl = LFT_IR;
		}
	}
}



/***** TO_LIGHT:  move towards brightest light *****/
void to_light(void)
{
	int i, j, scaleR, scaleL, irdr, irdl;

	scaleR = (photoR - 3) / 5;
	scaleL = photoL / 5;

	while(scaleR > scaleL)
	{
		turn_right();
		scaleR = (photoR - 3) / 5;
		scaleL = photoL / 5;
	}

	while(scaleL > scaleR)
	{
		turn_left();
		scaleR = (photoR - 3) / 5;
		scaleL = photoL / 5;
	}

	irdr = RGT_IR;
	irdl = LFT_IR;
	if((irdr < IRR_LMT) && (irdl < IRL_LMT))
		for (j=5; j>0; j--)
		{
			forward();
			for (i=0; i<10000; i++);
			avoid();
		}
	else
	{
		avoid();
		for (j=5; j>0; j--)
		{
			forward();
			for (i=0; i<10000; i++);
			avoid();
		}
	}
}



/***** FROM_LIGHT:  move away from light *****/
void from_light(void)
{
	int i, j, scaleR, scaleL, irdr, irdl;

	scaleR = (photoR - 3) / 5;
	scaleL = photoL / 5;

	while(scaleR > scaleL)
	{
		turn_right();
		scaleR = (photoR - 3) / 5;
		scaleL = photoL / 5;
	}

	while(scaleL > scaleR)
	{
		turn_left();
		scaleR = (photoR - 3) / 5;
		scaleL = photoL / 5;
	}

	turn_right();
	wait(750);

	irdr = RGT_IR;
	irdl = LFT_IR;


	avoid();
	for (j=25; j>0; j--)
	{
		forward();
		for (i=0; i<10000; i++);
			avoid();
	}
}




/***** POWER:  check battery voltage *****/
void power(void)
{
	int charge;

	charge = volts;
	while (charge < CHG_lmt)
	{
		turn_left();
		charge = volts;
	}
}



/***** WANDER:  wander, duh *****/
```

```
void wander(void)
{
        int i, j, time;
        unsigned rand;

        rand = TCNT;
        time = 0;

        avoid();
        forward();

        if (rand % 4)
                time = 0;
        if ((rand % 4) - 1)
                time = 300;
        if ((rand % 4) - 2)
                time = 600;
        if ((rand % 4) - 3)
                time = 900;

        if (rand & 0x0001)
                turn_left();
        else
                turn_right();
        wait(time);

        for (i=0; i<10000; i++)
        {
                avoid();
                forward();
        }
}



/***** DANCE:  you will dance for me *****/
void dance(void)
{
        int i, j, time;
        unsigned rand;

        rand = TCNT;

        if (rand % 4)
                time = 0;
        if ((rand % 4) - 1)
                time = 300;
            if ((rand % 4) - 2)
                time = 600;
            if ((rand % 4) - 3)
                time = 900;

        turn_left();
        wait(time);
        turn_right();
        wait(time);
        turn_left();
        wait(time);
        turn_right();
        wait(time);
        for (j=25; j>0; j--)
                for (i=0; i<10000; i++);
                {
                    forward();
                    avoid();

                }
        turn_right();
        wait(time);
        turn_left();
        wait(time);
        turn_right();
        wait(time);
        turn_left();
        wait(time);


        for (j=25; j>0; j--)
        {
                back();
                for (i=0; i<1000; i++);
                for (i=0; i<1000; i++);
        }
        turn_left();
        wait(time);
        turn_right();
        wait(time);
        turn_left();
        wait(time);
        turn_right();
        wait(time);
}
```

```
void five(void)
{
	int i,j;

	for(j=0; j<50; j++)
	{
		*(unsigned char *) 0x7000=(buz);
		for(i=0; i<100; i++);
		*(unsigned char *) 0x7000=(none);
		for(i=0; i<100; i++);
	}
	for(j=0; j<50; j++)
	{
		*(unsigned char *) 0x7000=0xD7;/*(lti || rti || buz);*/
		for(i=0; i<100; i++);
		*(unsigned char *) 0x7000=0xC7;/*(lti || rti);*/
		for(i=0; i<100; i++);
	}
	*(unsigned char *) 0x7000=0xC7;
}

void two(void)
{
	int i, j;

	for(j=0; j<4; j++)
	{
		for(i=0; i < 10000; i++)
			*(unsigned char *) 0x7000=0xD7;
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++)
			*(unsigned char *) 0x7000=0xC7;
	}
	for(j=0; j<4; j++)
	{
		for(i=0; i < 10000; i++)
			*(unsigned char *) 0x7000=0x97;
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++);
		for(i=0; i < 10000; i++)
```

```
                        *(unsigned char *) 0x7000=0xC7;
        }

        for(j=0; j<4; j++)
        {
                for(i=0; i < 10000; i++)
                        *(unsigned char *) 0x7000=0x57;
                for(i=0; i < 10000; i++);
                for(i=0; i < 10000; i++);
                for(i=0; i < 10000; i++);
                for(i=0; i < 10000; i++)
                        *(unsigned char *) 0x7000=0xC7;
        }
}



void one(void)
{
        int i,j;

        for(j=0; j<4; j++)
        {
                for(i=0; i < 500; i++)
                        *(unsigned char *) 0x7000=0x97;

                for(i=0; i < 5000; i++)
                        *(unsigned char *) 0x7000=0xC7;
                for(i=0; i < 500; i++)
                        *(unsigned char *) 0x7000=0x57;

                for(i=0; i < 5000; i++)
                        *(unsigned char *) 0x7000=0xC7;
        }
}


/*****************************************************/
/*****************************************************/
/******      ******************************************/
/***** States ******************************************/
/******      ******************************************/
/*****************************************************/
/*****************************************************/
```

```
void angry(void)
{
        int i, j, k, choice;
        unsigned rand;

        rand = TCNT;
        choice = rand % 9;

        if(choice==5)
        {
                for(k=0; k < 1; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;
                for (j=0; j<3; j++)
                        wander();
        }

        if((choice==6) || (choice==8))
        {
                for(k=0; k < 2; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;
                for (j=0; j<10; j++)
                        to_light();
        }

        if((choice==2) || (choice==4) || (choice==7))
        {
                for(k=0; k < 3; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;
                for (j=0; j<10; j++)
                        from_light();
        }

        if((choice==3) || (choice==0) || (choice==1))
        {
                stop();
                for (j=0; j<15; j++)
                        five();
        }
}



void disgruntled(void)
{
        int i, j, k, choice;
        unsigned rand;

        rand = TCNT;
        choice = rand % 7;

        if(choice==5)
        {
                for(k=0; k < 1; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;
                for (j=0; j<3; j++)
                        wander();
        }

        if((choice==3) || (choice==0))
        {
                for(k=0; k < 4; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;

                forward();

                for (i=0; i<30; i++)
                {
                        *(unsigned char *) 0x7000=0xC7;
                        for(j=0; j<30000; j++);
                                avoid();
                        *(unsigned char *) 0x7000=0x07;
                        for(j=0; j<30000; j++);
                                avoid();
                }
        }

        if((choice==1) || (choice==6))
        {
                for(k=0; k < 3; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;

                for (j=0; j<5; j++)
                        from_light();
        }

        if((choice==2) || (choice==4))
        {
                stop();
                for (j=0; j<15; j++)
                        five();
        }

}




void pleased(void)
```

```
{
        int i, j, k, choice;
        unsigned rand;

        rand = TCNT;
        choice = rand % 6;

        if((choice==0) || (choice==1) || (choice==5))
        {
                for(k=0; k < 2; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;

                for (j=0; j<10; j++)
                        to_light();
        }



        if(choice==2)
        {
                for(k=0; k < 1; k++)
                {
                        *(unsigned char *) 0x7000=(buz);
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;

                for (j=0; j<5; j++)
                        wander();
        }

        if((choice==3) || (choice==4))
        {
                stop();
                two();
        }
}
```

```
void happy(void)
{
	int i, j, k, choice;
	unsigned rand;

	rand = TCNT;
	choice = rand % 11;

	if((choice==0) || (choice==6) || (choice==8) || (choice==10))
	{
		for(k=0; k < 2; k++)
		{
			*(unsigned char *) 0x7000=(buz);
			for(i=0; i<10000; i++);
			*(unsigned char *) 0x7000=(none);
			for(i=0; i<10000; i++);
		}
		*(unsigned char *) 0x7000=0xC7;

		for (j=0; j<30; j++)
			to_light();
	}

	if(choice==1)
	{
		for(k=0; k < 1; k++)
		{
			*(unsigned char *) 0x7000=(buz);
			for(i=0; i<10000; i++);
			*(unsigned char *) 0x7000=(none);
			for(i=0; i<10000; i++);
		}
		*(unsigned char *) 0x7000=0xC7;

		for (j=0; j<5; j++)
			wander();
	}

	if((choice==2) || (choice==9))
	{
		for(k=0; k < 5; k++)
		{
			*(unsigned char *) 0x7000=(buz);
```

```
                        for(i=0; i<10000; i++);
                        *(unsigned char *) 0x7000=(none);
                        for(i=0; i<10000; i++);
                }
                *(unsigned char *) 0x7000=0xC7;

                for (j=0; j<2; j++)
                        dance();
        }



        if((choice==3) || (choice==5) || (choice==7))
        {
                stop();
                one();
        }

        if(choice==4)
        {
                stop();
                two();
        }
}
```