Table of Contents

Abstract

Canary is an autonomous rolling sensor platform designed specifically to detect, neutralize, and alert humans to the presence of toxins in the air. In lieu of using true airborne toxins, Electro Magnetic Field sensors were used as a simulation. Canary discovers an EMF, then follows the emanations to a danger point. Canary then releases an anti toxin, and monitors the scene for human approach. If humans approach, Canary will sound an alarm to warn them of danger. Through the use of bump, IR, EMF, and motion sensors, Canary will protect lab users from hazardous, and undetectable (to humans) gases that can leak and cause serious injury.

Executive Summary

Canary is named for the coal miner's canaries of old, who would warn miners of toxic and undetectable gases by, unfortunately, dying. Canary is an attempt to keep the function of the canaries of old, but to avoid the death of birds, and also to enable quick-response measures to be taken to eliminate the threat, or at least warn humans of potential danger.

Canary is built on a Talrik Junior platform, and as such is compact and agile. Canary will wander about a lab, avoiding obstacles and constantly checking it's sensors for potential harmful chemicals in the air. Once a chemical is detected, Canary will try to find the source of the disturbance. Once the source is found, an anti toxin or some other reaction to try to fix the leak will be performed. Also, Canary will monitor the area and if humans venture too close, an alarm swill sound, warning them of danger.

In lieu of toxic fume detectors, EMF detectors are used to simulate an airborne toxin. By creating a modular sensor array, Canary can easily be fitted with various sensor arrays specific for the lab or application of the user. An onboard speaker will sound a siren if humans approach a leak, and Canary is equipped with visual and well as audio warnings to facilitate use with hearing impaired users, or in high-noise areas.

Canary's purpose is straightforward: save lives. Many of the hazards in a typical laboratory are invisible, and many cannot be detected by humans in any way short of inhaling enough of it to suffer the consequences. Canary strives to give warning and to provide a first line of defense against toxic airborne chemicals.

INTRODUCTION

Canary is a robot designed to perform consistent, constant "sniffing" for potentially dangerous materials in the air. To this end, a sensor array will "sniff" the air and Canary will follow the greatest concentration to its center. Then, Canary will alert any humans nearby about the potential danger so that it can be avoided or remedied. Using a Talrik Junior rolling robot to mount the sensor array and to navigate around obstacles, Canary will ease researcher's minds and technician's hearts as it finds invisible leaks and spills and alerts the proper authorities.

MOBILE PLATFORM

The mobile platform needs to be compact, so as to be out of the way of everyday workers. Walking is not required, so I decided on a rolling platform. The sensors and other equipment are not heavy, and no additional power is needed. With these specifications in mink, the Talrik Junior platform is the one I am using. The ease of construction, and familiarity of TAs with the design is essential to get a reliable platform quickly.

The large flat surface on top of the Talrik is an ideal platform to mount the sensor array. The circular bumper ensures that Canary can move about and not get stuck (pending coding). The size of the Talrik will also make Canary unobtrusive, and therefore a more desirable facet of corporate life.

ACTUATION

The actuation for Canary is straightforward and well tested. By using the Talrik Junior, and motors supplied by Tektronix, I am using actuation that has been used by several people before me. This satisfies me, as I am not very qualified, nor interested, in creating a novel platform. I want to proceed quickly to other levels of robot design, and a dual servomotor rolling platform allows me to move onto the electronics and behaviors quickly.

Sensor Scope and Objectives

Canary's sensors are designed to allow the robot freedom of movement around a laboratory, and detect toxins. The sensors are also designed to allow Canary to find the highest concentration of a detected toxin, and warn humans of potential danger. To this end, IR and bump sensors are used to allow freedom of movement, a single EMF detector detects the "toxin", an array of EMF sensors will allow Canary to find the highest concentration of the toxin, and a motion sensor will allow Canary to warn humans of potential danger.

Bump Sensor

Bump sensors are used as a last ditch sensor to determine when obstacles are in front of Canary. Canary is equipped with four such sensors, switches that are triggered when the ring bumper impacts an obstacle. These switches are either on or off, and will change the behavior of Canary when activated to avoid the obstacle detected. Bump switches are also used on Canary to reset the robot and make him begin searching for toxins after a toxin is eliminated.

Bump Sensors have been used by almost every robot in IMDL for some time. Their use and value is well documented and tested. These inexpensive switches provide Canary with a limited sense of touch that is invaluable when other sensors fail.

IR Sensor

Infrared Sensors are mounted on Canary and are it's "eyes". The emitters emit Infrared light, and the hacked IR cans receive this light after it is reflected off of objects near Canary. The IR cans have been hacked to provide analog values instead of digital ones. The two IR cans when used together will allow Canary to decide which way to turn in the event of an obstacle presenting itself.

These sensors have also been made use of extensively by IMDL robots in the past, and have a well documented behavior, which is dependent on the emitter and placement of the IR can. Also, these sensors are sensitive to the object that is reflecting the emitted light. Dark colors and some textures will render the IR sensor useless.

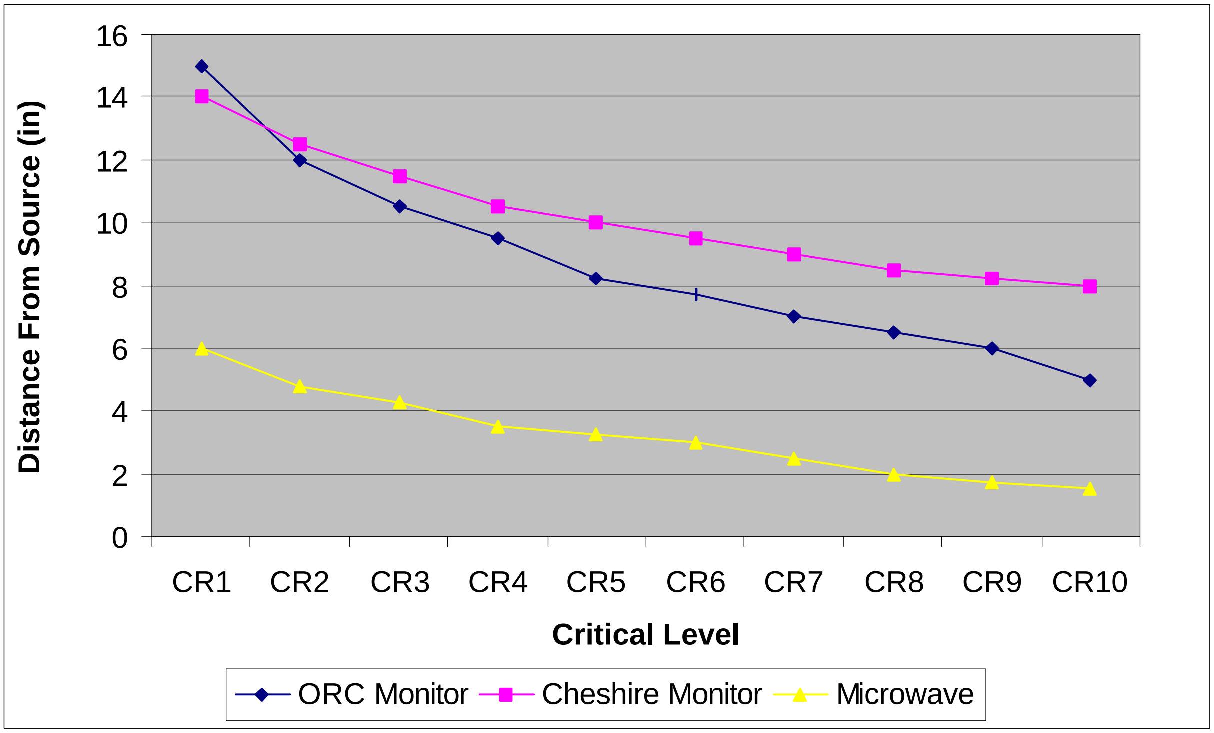
EMF Sensor

The electro-magnetic-field sensor is a large receiver hooked up to several chips to provide an analog output of relative magnetic field strength. This sensor is being used because of it's cost-effectiveness, and because EMF is not toxic to humans, a preferable "toxin" to look for in demonstrations and debugging than any real toxic airborne chemical.

The sensors originally gave data based on 10 LEDs, increasing the number of LEDs lit as the strength of the Electro Magnetic field increased. With the help of the TAs, I hacked these sensors (actually a very easy hack as the chip prior to the LEDs merely took an analog input and converted it into 10 LED outputs) for use on Canary.

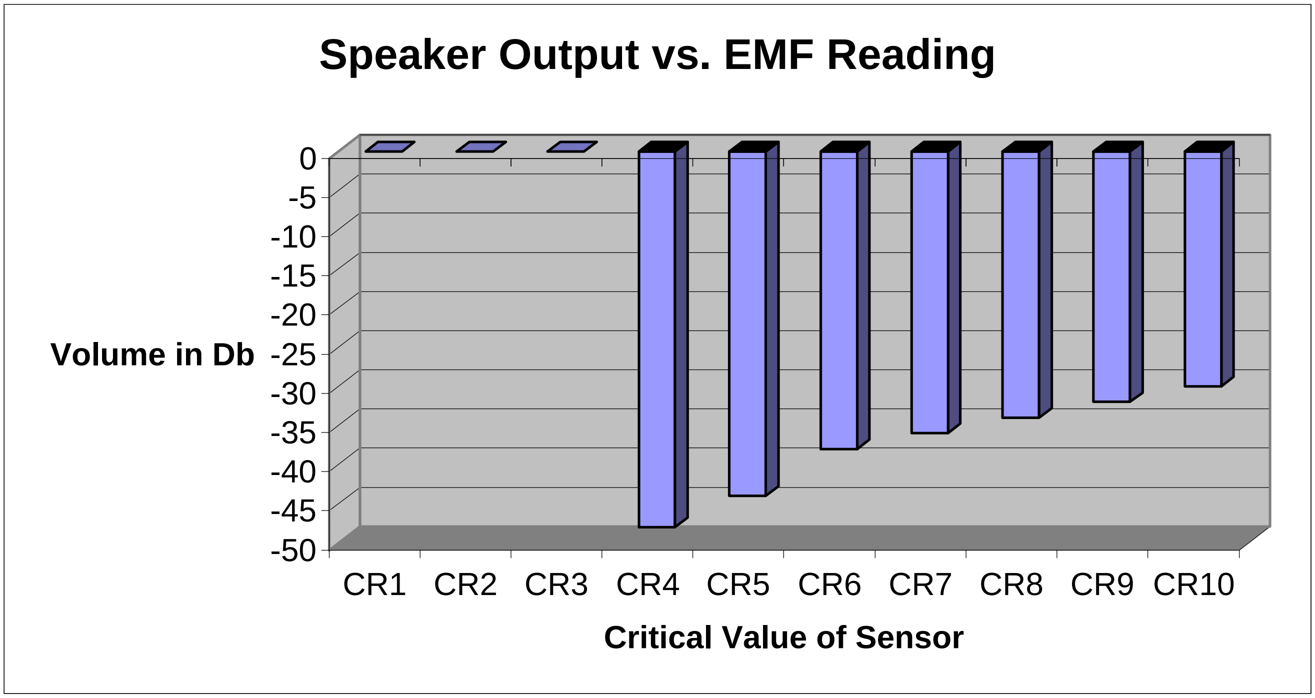
The EMF sensors will be placed in a 3 sensor array, so that differential data can be used. This way, Canary can "follow" the scent of a high EMF concentration and find the highest concentration of the "toxin". This is useful because eventually Canary will want to release an antitoxin, and release in the highest toxin concentration is most favorable for effective anti toxin usage. Canary will NOT look for differentials in the 3rd dimension, as it lacks any means to raise itself above the floor.

Some home made experiments (done before the sensors were hacked) are included below. Using the LED critical points, several measurements were taken based on EMF emissions from various sources.



ORC monitor is a 15" 2 year old monitor, Cheshire monitor is a 17" 1 year old monitor. I was surprised that the monitors gave off more EMF than the microwave, as the microwave readings were from the crack between the door and housing of the microwave.

In addition to distance measurements, which are important for Canary's spatial alignment, I took readings at a distance of 1" from my home stereo's speakers. Based on the Db level given by the stereo, the following data was gathered:



The speaker output caused a minimum of CR4 when the stereo was on. This can be a good sign, as a strong field can be created by a speaker, which may be better in testing than monitors or other EMF emitters (as long as the judges like the music played that is).

Motion Sensor

The motion sensor is responsible for detecting movement of humans and sounding an alarm if Canary is following a toxin. This way, humans in the environment will know that something is wrong and will be able to stay away and call for more assistance. This is the most important part of Canary's function, as it enables the robot to potentially save people's lives.

A motion sensor that was hacked off of an outdoor light array is used on Canary. This inexpensive motion sensor works very well, detecting movement up to 10 feet away, and it’s range of vision is also very good, about 130 degrees.

BEHAVIORS

As a CISE major, I hope to do most of my innovation here. First is the collision avoidance behavior. The standard collision avoidance will be fine as long as Canary is in the “roam” mode, without having detected any EMF. However, if the EMF sensor array does find something, Canary does not necessarily want to merely avoid obstacles. There is a possibility that the readings are emanating FROM the obstacle, and the search and alert behavior may need to override the collision avoidance in order to pinpoint the EMF source.

My Search and Alert behavior will find the highest concentration of EMF and try to find a human close by. If a human is found, then a warning siren/beacon/whistle will go off, alerting the human to a potentially dangerous EMF.

The anti toxin release behavior executes after Canary has found what it considers a “danger” concentration. In this case, no remedy for EMF can be released, so a siren is used to simulate the release of the anti toxin.

Experimental Layout and Results

The experiments to evaluate Canary are very straightforward. A small room was constructed of reflective walls, and an EMF source (in this case a monitor) was placed in one wall. Canary is powered up, then will roam randomly until its EMF sensors detect the monitor. At that point, the following routine should take over, and Canary follows the EMF into the monitor. When it gets close enough to the monitor, Canary will release the anti – toxin. Then, Canary will spin about and look for humans to warn.

Testing was done in the IMDL lab to begin with. This kept Canary close to the serial interface, and allowed use of debugging techniques involving a PC. The code was well thought out, and a minimal amount (4 or 5 hours) of debugging produced code that would perform the tasks described above.

Several problems were encountered, causing a removal of 1 EMF detectors. The speaker to sound the alarm produced its own EMF that was sending readings off the scale. The speaker was moved from inside the sensor column to the rear of the vehicle (the speaker magnet was taken advantage of; I let it stick to the batteries). Then, after several attempts, I discovered that the analog ports were not mapped correctly to the EMF detectors. Once that was fixed, Canary performed well.

Conclusion

In conclusion, Canary performs its duties very well. The collision avoidance, EMF following, human warning, and anti toxin release behaviors all worked separately and in concert. I look forward to taking Canary's application from the EMF range to a more useful sensor array. In addition, future work could allow Canary to use the motion sensor to avoid humans while traveling about the lab, as a motion sensor's range is much greater than the IR range.

A new platform may be in order, as the weight and size of an antitoxin was not included in this version of Canary. In addition, other sensors (spill and pH for example) could be added to the bottom of Canary to detect liquid as well as gaseous toxins.

If I were to start this project over again, I do not feel that I would make too many changes. The scope of the project was attainable in a single semester, and the function of Canary is useful and interesting. I think I would have developed a more modular sensor array, and I would have cleaned up Canary's appearance. Beyond that, future work can remedy those ruminations, and additional function and aesthetics can be implemented.

Appendix - Source Code

```
/*************************** Includes ********************************/
#include <tjpbase.h>
#include <stdio.h>
/************************ End of Includes ****************************/



/*************************** Defines *********************************/
#define DEBUG 1
#define LEMF analog(5)
#define REMF analog(6)
#define FEMF analog(7)
#define BEMF analog (4)
#define IR_THRESHOLD 110
#define SLOW_TURN_SPEED 25
#define MOTION_SENSOR 100
#define EMF_THRESHOLD 120
#define MAX_TOXIN_THRESHOLD 230
#define IR_LOW_THRESHOLD 100
/************************ End of Defines *****************************/
int EMF_detected = 0;
int max_toxin = 0;
int no_antitoxin_loose = 1;
int motion_detected = 0;
int f_EMF = 0;
int r_EMF = 0;
int l_EMF = 0;
int b_EMF = 0;
int lf_bump = 0;
int rf_bump = 0;
int cf_bump = 0;
int rear_bump = 0;
int irdr = 0;
int irdl = 0;


void turn(void);
void sound_alarm(void);
void read_move_sensors(void);
void read_EMF_sensors(void);
void follow_toxin(void);
void collision_avoid(void);
void face_up(void);
void release_antitoxin(void);
void read_motion_sensors(void);


void main(void)
{
 init_analog();
  init_motortjp();
  init_clocktjp();
```

```
   IRE_ON;     /* turn on IR emitters */
while (1)
{

read_move_sensors();

read_EMF_sensors();
if (EMF_detected)
      {
      if (DEBUG) printf("follow_toxin\n");
      follow_toxin();
      if (max_toxin)
            {
            if (DEBUG) printf("face_up\n");
            face_up();
            while (!FRONT_BUMP)
                  {
                  motorp(RIGHT_MOTOR,0);
                  motorp(LEFT_MOTOR,0);
                  if (no_antitoxin_loose)
                        {
                              if (DEBUG) printf("release_antitoxin\n");
                        release_antitoxin();
                        no_antitoxin_loose = 0;
                        }     //end if (no_antitoxin_loose)
            //          if (DEBUG) printf("read_motion_sensors\n");
                  read_motion_sensors();
                  if (motion_detected)
                        {
                        if (DEBUG) printf("sound_alarm\n");
                        sound_alarm();
                        }     //end of if(motion_detected)
                  }     //end of while(!BUMP)
            }     //end of if(max_toxin)
      }  //end if (EMF_detected)
else
      {
      if (DEBUG) printf("collision_avoid+\n");
      collision_avoid();
      }     // end of else

}           // end while (1)

}     // end of main

void sound_alarm(void)
{
      SET_BIT(PORTA,0x20);
      wait(2000);
      CLEAR_BIT(PORTA,0x20);
      wait(2000);
}

void read_move_sensors(void)
{

lf_bump = FRONT_BUMP;
```

```
if(DEBUG) printf("LFB = %d\t",lf_bump);
rf_bump = FRONT_BUMP;
if(DEBUG) printf("RFB = %d\t",rf_bump);
cf_bump = FRONT_BUMP;
if(DEBUG) printf("CFB = %d\t",cf_bump);
rear_bump = BACK_BUMP;
if(DEBUG) printf("CBB = %d\t",rear_bump);
irdr = RIGHT_IR;
if(DEBUG) printf("RightIR = %d\t",irdr);
irdl = LEFT_IR;
if(DEBUG) printf("LeftIR = %d\n",irdl);



}      //end of read_move_sensors()

void read_EMF_sensors(void)
{
EMF_detected = 0;
f_EMF = FEMF;
if(DEBUG) printf("FEMF = %d\t",f_EMF);
r_EMF = REMF;
if(DEBUG) printf("REMF = %d\t",r_EMF);

l_EMF = LEMF;
if(DEBUG) printf("LEMF = %d\t",l_EMF);
b_EMF = BEMF;
if(DEBUG) printf("BEMF = %d\n",b_EMF);
if(f_EMF >= EMF_THRESHOLD || r_EMF >= EMF_THRESHOLD || l_EMF >=
EMF_THRESHOLD)
      EMF_detected = 1;

}      //end of read_EMF_sensors()

void follow_toxin(void)
{

int right_power = 0;
int left_power = 0;
int EMF_direction_right = 0;
int EMF_direction_left = 0;
int IR_direction_right = 0;
int IR_direction_left = 0;
int max_EMF = 0;

if (f_EMF >= b_EMF)
{
      EMF_direction_right = 100;
      EMF_direction_left = 100;
}

if (l_EMF >= r_EMF)
{
      EMF_direction_right = EMF_direction_right + 50;
      EMF_direction_left = EMF_direction_left - 50;
}
```

```
if (r_EMF > l_EMF)
{
      EMF_direction_right = EMF_direction_right - 50;
      EMF_direction_left = EMF_direction_left + 50;
}



if(irdl > IR_THRESHOLD)
{
IR_direction_right = IR_direction_right + 100;
IR_direction_left = IR_direction_left - 100;
}

if(irdr > IR_THRESHOLD)
{
IR_direction_right = IR_direction_right - 100;
IR_direction_left = IR_direction_left + 100;
}


// right_power = (EMF_direction_right * .33) + (IR_direction_right *
.66);
// left_power = (EMF_direction_left * .33) + (IR_direction_left * .66);

if(FRONT_BUMP) turn();
right_power = EMF_direction_right;
left_power = EMF_direction_left;

if (right_power > 100) right_power = 100;
if(left_power > 100) left_power = 100;
if (right_power < -100) right_power = -100;
if (left_power < -100) left_power = -100;

motorp(RIGHT_MOTOR, right_power);
motorp(LEFT_MOTOR, left_power);

 max_EMF = l_EMF;
// if (b_EMF > l_EMF )max_EMF = b_EMF;
if (r_EMF > l_EMF )max_EMF = r_EMF;
if (f_EMF > l_EMF )max_EMF = f_EMF;

// if (irdl > IR_THRESHOLD || irdr > IR_THRESHOLD) max_EMF = max_EMF +
30;

if (max_EMF > MAX_TOXIN_THRESHOLD) max_toxin = 1;

}     //end of follow_toxin()

void collision_avoid(void)
{
      int speedr = 0;
      int speedl = 0;

    if (irdl > IR_THRESHOLD)
      speedr = -MAX_SPEED;
    else
```

```
      speedr = MAX_SPEED;
    if (irdr > IR_THRESHOLD)
      speedl = -MAX_SPEED;
    else
      speedl = MAX_SPEED;

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


if(cf_bump || lf_bump || rf_bump)
    {
      motorp(LEFT_MOTOR, -MAX_SPEED);
      motorp(RIGHT_MOTOR, -MAX_SPEED);
      wait(600);
      turn();
    }

} // end of collision_avoid()

void face_up(void)
{
while(irdl >= IR_LOW_THRESHOLD || irdr >= IR_LOW_THRESHOLD)
      {
      irdl = LEFT_IR;
      irdr = RIGHT_IR;
      motorp(RIGHT_MOTOR,SLOW_TURN_SPEED);
      motorp(LEFT_MOTOR,-SLOW_TURN_SPEED);
      }     //end of while (irdl >= IR_LOW_THRESHOLD...)
}     // end of face_up()

void release_antitoxin(void)
{
      SET_BIT(PORTA,0x20);
      wait(2000);
      CLEAR_BIT(PORTA,0x20);
      wait(2000);
      SET_BIT(PORTA,0x20);
      wait(2000);
      CLEAR_BIT(PORTA,0x20);
      wait(2000);
}     // end release_antitoxin

void read_motion_sensors(void)
{
motion_detected = !(PORTA & 0x01);
}     // end read_motion_sensors



void turn(void)
{
 int i;
 unsigned rand;
if (DEBUG) printf("turning");
 rand = TCNT;
 if (rand & 0x0001)
 /*turn left*/
 {
  if(DEBUG) printf(" left\n");
   motorp(RIGHT_MOTOR, MAX_SPEED);
   motorp(LEFT_MOTOR, -MAX_SPEED);
 }
 else
 /*turn right*/
 {
 if(DEBUG) printf(" right\n");
   motorp(RIGHT_MOTOR, -MAX_SPEED);
   motorp(LEFT_MOTOR, MAX_SPEED);
 }

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

}
```