IMDL Fall 2010
Final Report

Robot Spotting

Courtney Hayes

Summary
This robot project consists of a large robot and a smaller one interacting with each other. The larger robots objective is to track another smaller mobile one. When the larger robot has a “lock” on the smaller robot it should then attempt to turn it it off by moving towards the robot and activating one of its sensors.

Hardware Overview
The large robot is OWI 007 educational robotic arm setup to be programmed for original behavior.
The large robot has 5 points of articulation: gripper, wrist, elbow, shoulder and base.

Illustration 1: Robotic Arm

Each has its own 3V DC motor. These motors are driven by two Toshiba TB6612FNG dual 1A motor drivers. The Pridgen Vermeer Robotics board controlled motor logic while a separate 3.3 V lithium supply sourced the motors. A PIR sensor and CMU cam is also interfaced with the PVR and were mounted on the arm.

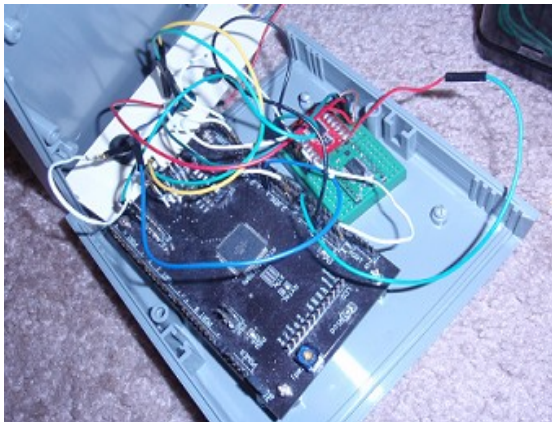

Illustration 2: PVR board and motor drivers

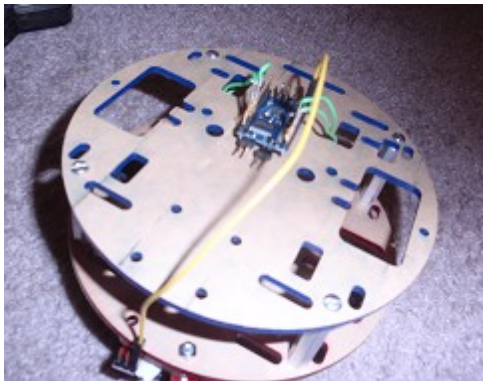

Illustration 3: Smaller robot with Baby Orangutan controller

The smaller robot is a two level circular platform. The bottom uses a Tamiya gearbox with a 203:1 low speed gear ratio to control two wheels. Motors and sensors are controlled by the "Baby Orangutan" a mini Arduino platform with two built in motor drivers also based on the Toshiba TB6612FNG. LEDs on the board are used for debugging. It has two primary sensors, reflectance and CDS cells.

Sensors

Pololu Reflectance sensor

The QT-RC is a digital output reflectance sensor
This sensor has two infrared leds. When a supply voltage is provided to the sensor it will return a value close to 0 when placed in front of a white surface and a value close to Vin in front of black surface.
The QT-1A is an analog sensor that returns a value from 0 to 1000 based on the amount a reflectance from an encountered surface.
Two of these sensors were mounted on the on the front ball caster of the robot. Three were originally planned for accuracy. With two sensors it is possible to straddle a line.

Parallax PIR sensor

PIR senses for sudden changes in the infrared field around it. The PIR sensor is polled at the end of the color tracking loop to detect movement.

CDS cell

CDS cell was to be used as a light sensitive resistor. When the robotic arm hovered over the cell the robot should halt.

CMU cam

The CMU cam is a camera that can be easily interfaced with a micro controller through its RS-232 ports or TTL. Its best use is to track bright primary colors. It was used to track a colored object mounted on a smaller robot.

Behavior
The object to be tracked is placed in front of the robotic arm and a command is sent to the camera to record the color in the center of the camera's screen. With the color parameter stored a track color command is sent to the camera. The robotic arm then receives centroid and confidence data from the camera and turns accordingly. When the robot sees “center” for more than three loop iterations the shoulder joint is bent down to block light from the CDS cell on the smaller mobile robot.

Due to loss of a sensor and time constraints The smaller mobile robot was designed to use two reflectance sensors to straddle a line. This was reduced from a more accurate three sensor array to track a black line.
When the smaller robot sensed a change in the CDS sensor it stopped

Issues
A large amount of time was spent on getting the CMU cam to work consistently. When attempting to control the shoulder joint the camera would frequently not turn on its tracking light. A single line a code added in the main while loop would seem to affect camera initialization outside of the loop.

Also with only two reflectance sensors the small mobile robot would frequently lose its line making it unsuitable to demo. It was difficult to determine whether motor power was being drawn from the PVR or the 3V battery. Occasionally the camera and board would fail unpredictably and drain the PVR battery

Because of this CDS sensor was not properly implemented in the design

Conclusion
Overall this project was a great learning experience, but far from a success. It introduced me to the functioning of motors, relays and drivers as forms of robot control. I discovered how difficult it is to predict the functioning of sensors and the amount of careful work needed to interface with a micro-controller. Some major mistakes I made is not getting my special sensor earlier and changing too much hardware and behaviors towards the end of the semester. For demo day I should have relied on sensors such as the ultrasonic distance detector to demo the mobile robot since I had worked out most of the problems in the early in the semester. Without the help and suggestions of the TAs and professors to project would have become more difficult than it was.

Appendix

Code for Arm

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

void CMUsend(char *command)
{
int i = 0;
  while (command[i] != '\0') //While command does not end do...
   {
            // Data Register Empty Flag: check if data register is empty
            while (!(USARTE0_STATUS & (1<<USART_DREIF_bp)));
            //USARTE0_DATA is shared by the transmit and receive
            USARTE0_DATA = command[i];
            //if data is sent TXCIF is set
            while (!(USARTE0_STATUS & (1<<USART_TXCIF_bp)));
            i++;
            }

}


char CMUreceive(void)
{
int i=0;
static char *data;
            do
            { // wait for receive is complete
            while (!(USARTE0_STATUS & (1<<USART_RXCIF_bp))){} //Put data into string
            data[i] = USARTE0_DATA;
            } while (data[i++] != '\r'); //Since all trasnfers are ended by \r wait for it to happe
            return data;
}

void main(void)
{
            xmegaInit();                                                    //setup XMega
            delayInit();                                                    //setup delay functions

            lcdInit();
            //lcdString("W");                           //display "PV Robotics" on top line (Line 0) of LCD
            lcdGoto(0,0);                                                   //move LCD cursor to the
second line (Line 1) of LCD

            PORTQ_DIR |= 0xFF;
            PORTF_DIR |= 0xFF;lude "PVR.h"
#include <stdlib.h>
#include <stdio.h>

   PORTE_DIR = PIN3_bm; //Pin 3 of port E is output
   PORTE_OUT = PIN3_bm; //Pin 3 of port E is TXO
   PORTE.DIRCLR = PIN2_bm; //Pin 2 of port E is RXO
   USARTE0_CTRLC = 0x03; // USART Control Register C: ASYNCHRONOUS, no parity 1 stop bit 8 bit word
   USARTE0_BAUDCTRLA = 0x06; // Page 238 of Atmel manual, fbaud=115200 (my case)=32MHz/
(16*(((2^BSCALE) * BSEL)+1))
   USARTE0_BAUDCTRLB = 0xC1; // BSEL= 262 and BSCALE= -4 in 2s comp .
   USARTE0_CTRLB |= 0x08; // TX0 is on
   USARTE0_CTRLB |= 0x10; // RX0 is on
```

```
static char *temp;
int last = 0;
int mov = 0;
//Program

CMUsend("RS\r"); //Reset the camera
delay_ms (5000); //Long enough to wait for ACK and to get the camera ready

CMUsend("L1 1\r"); //To turn on green light
temp = CMUreceive(); //Receive ACK
delay_ms(5); //For some reason that I havent figure out, there must be a delay between commands
//lcdString(temp);

CMUsend("PM 1\r"); //Activate polling mode
temp=CMUreceive(); //Receive ACK
delay_ms(5);
//lcdString(temp);

CMUsend("CR 18 32 19 32\r"); // Turn off auto gain/WB
temp=CMUreceive(); //Receive ACK
delay_ms(5);

CMUsend("TW\r"); //Track window
temp=CMUreceive(); // Receive ACK
temp=CMUreceive(); // Receive S and M packet
delay_ms(500);

while(1)
{

CMUsend("TC\r"); //Track window
temp=CMUreceive(); // Receive ACK
temp=CMUreceive(); // Receive M packet
//p = strlen(temp);  // code for lcd debugging
//lcdGoto(0,0);
//lcdChar(temp[2]);  // receives center value
//lcdGoto(0,2);
//lcdInt(p);
//lcdChar(temp[16]); // receives confidence
//lcdChar(temp[17]);
//lcdChar(temp[18]);
delay_ms(20);

if(temp[2] < 52)
  {
  delay_ms(75);
  PORTF_OUT =0x0F;
  delay_ms(10);
  PORTF_OUT = 0x01;;
  PORTQ_OUT = 0x00;
  //delay_ms(1500);
  delay_ms(5);
//  PORTF_OUT =0x04;
  }
else if(temp[2] > 52)
{
delay_ms(75);
PORTF_OUT = 0x0F;
delay_ms(10);
PORTF_OUT = 0x04;
//delay_ms(1500);
```

```
PORTQ_OUT = 0x01;
delay_ms(5);

}
else
{
 delay_ms(75);
 PORTF_OUT = 0x00;
// if(mov > 3) This detected if center was seen more than three times
 //{
  //delay_ms(75);
  //PORTF_OUT = 0x80;
  //delay_ms(1500);
  //PORTF_OUT = 0x40;
  //delay_ms(1500);
  //PORTF_OUT = 0x00;
  //mov = 0;
 // }
  //mov++;
 }

}

}
```

Code for Small Mobile Robot

```
#include <pololu/orangutan.h> //header for board functions

int main()
{
 // initialize analog and digital QTR sensors
 unsigned char qtr_rc_pins[] = {IO_C1};
 qtr_rc_init(qtr_rc_pins, 1, 2000, 255);  // 800 us timeout, no emitter pin
 int qtr_analog_pins = {0}
 qtr_analog_init(qtr_analog_pins, 1, 10, IO_C0); // 10 samples, emitter pin is PC0
 unsigned int sensors[2];
 BOOL touch = TRUE;

 for (int i = 0; i < 250; i++)  //calibrate for about 5 seconds
 {
  qtr_calibrate(QTR_EMITTERS_ON);
  delay(20);
 }

while(touch)
{

 int position = qtr_read_line(sensors, QTR_EMITTERS_ON);

 int error = position - 1000;

 int leftMotorSpeed = 100;
 int rightMotorSpeed = 100;
 if (error < -500)  // the line is on the left
  leftMotorSpeed = 0;  // turn left
 if (error > 500)  // the line is on the right
  rightMotorSpeed = 0;  // turn right
 if( IO_C2 > 256)
  {
    touch = TRUE
```

```
    }
}
```