Sensor Report

EEL 5666
Intelligent Machine Design Lab

Sara Keen
March 17, 2005

Table of Contents

Introduction

My two robots, Zack and A.C. will be equipped with numerous sensors that will enable them to perform intelligently in any environment. Ultimately all of the sensors working together will permit both robots to "see" and understand what is happening around them and their partner. This report discusses the four sensors I am using and their purposes.

Infrared Sensors

I am using 4 Sharp GP2D12 Infrared distance sensors to employ obstacle avoidance. Each sensor requires a JST three pin connector to be interfaced with the microcontroller.

Theory of Operation

While the GP2D12 is connected to ground and power the sensor takes continuous distance readings and outputs the result as an analog voltage. I connected the GP2D12s directly to the pins for the analog-to-digital converter on my microcontroller to obtain digital results. The sensors have a range of approximately 4 to 30 inches. Following is a chart showing readings from all four sensors at various distances.

Dist (in)	IR1	IR2	IR3	IR4
1	190	188	172	172
2	351	328	347	332
3	495	485	517	505
4	508	456	456	511
5	427	380	388	403
6	333	317	328	320
7	277	281	291	284
8	246	251	260	256
9	223	226	234	233
10	205	208	214	214
11	196	196	200	198
12	182	180	182	178
13	162	161	164	164
14	151	150	157	155
15	141	146	149	144
16	134	133	142	140
17	125	130	133	132
18	122	126	125	123
19	126	119	121	115
20	122	113	122	111
21	118	106	109	107
22	113	101	105	103
23	109	98	102	96
24	102	94	97	92

Software Implementation

It is obvious from the above chart that all of the sensors are nearly identical, therefore my code need not acknowledge which sensor the robot is reading from. At extremely close distances readings tend to be inaccurate, and my code accounts for this by beginning to turn when an obstacle is about ten inches away. I used the main clock to create interrupts every millisecond to take readings from the sensors. A polling routine can not count the time before an infrared signal is echoed back because any interrupt could pause the polling routine and render all of the readings. Using timer interrupts the analog to digital converter takes five readings every millisecond and returns their average. The following code shows the timer interrupt initializations and ADC calculations.

```
void init_timer0(void)
{
  TCCR0 = 0;
  TIFR  |= BV(OCIE0)|BV(TOIE0);
```

```
  TIMSK |= BV(TOIE0)|BV(OCIE0);         /* enable output compare interrupt */
  TCCR0  = BV(WGM01)|BV(CS02)|BV(CS00); /* CTC, prescale = 128 */
  TCNT0  = 0;
  OCR0   = OCR_1MS;                    /* match in aprox 1 ms */
}

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();
  }
```

Using the constant readings from the GP2D12s it was easy to implement an obstacle avoidance routine. Here is an example of how the robot would search for and react to an obstacle on the left.

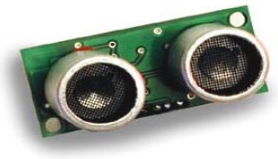
```
while (1) {

 //read left sensor
    irleft = adc_readn(2, 5);  /* sample channel 5 times, take average */

//read right sensor
  irright = adc_readn(3, 5);  /* sample channel 5 times, take average */

 if (( irleft > 200 ) & ( irright < 200 )) {
          if ( irflag != 1) {                          //if this is the first detection

    clr_lcd();
    printf("Obstacle on left");

          irflag = 1;
          SERVO5 = SERVO_FOR4;              // left servo faster than
          SERVO6 = SERVO_FOR2;              // right servo
          }
```

```
        }
}
```

As my robots are small and will not be moving at high speeds, only two sensors are required for each robot. The sensors are placed on the front corners of the platform facing approximately 20° outward. This allows plenty of warning before collision occurs.

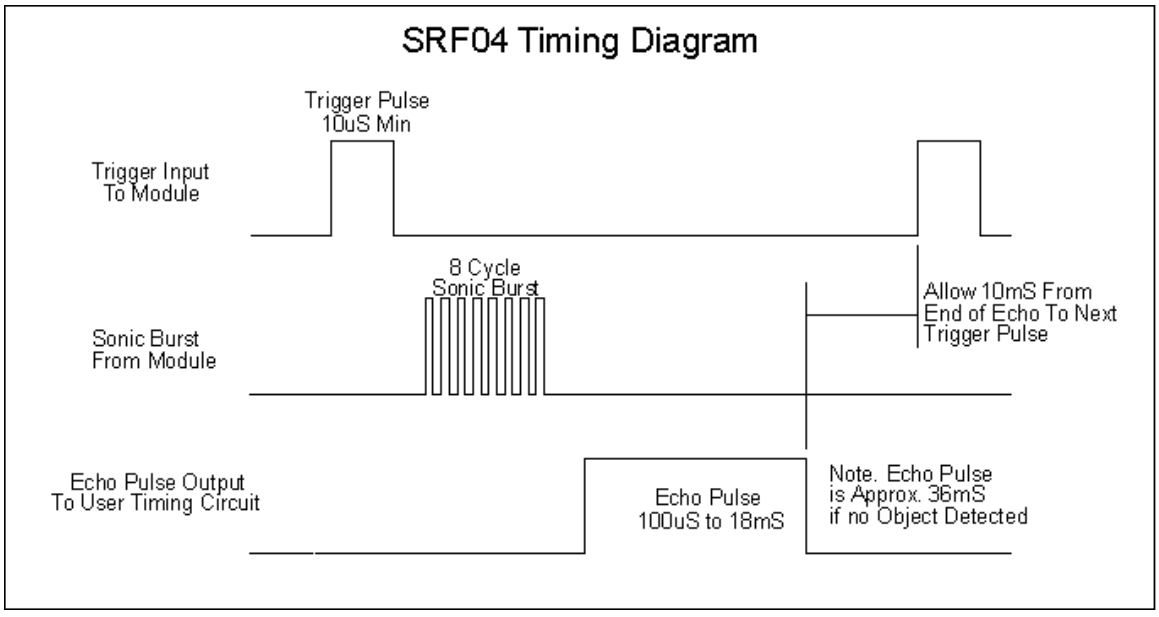
Ultrasonic Sensors

Both robots will be equipped with two Devantech SRF04 Ultrasonic rangefinders to position themselves in front of objects they will pick up. With two sensors that have a known angle between them a robot can easily determine perpendicular distance.



Theory of Operation

These sensors can be controlled by using the timers and normal i/o pins on the microcontroller. To begin a reading a signal to the trigger input of the SRF04 must be held high for at least 10 us. At the falling edge of the trigger the ultrasonic ping is emitted. After about 100us the microcontroller begins listening for the echo using an input pin. The timing diagram of the SRF04 is shown below.



The table below contains readings from all four sensors. Note that the readings are not representative of distance, they simply represent the number of delays that occurred before an echo was received. To detect nearby objects the robots look for readings less than a certain value.

Dist (in)	1	2	3	4
1	15	17	15	15
1.375	14	16	18	14
1.625	17	18	20	17
2	21	24	22	21
2.375	22	25	26	22
2.625	26	29	33	26
3	30	34	38	30
3.375	34	39	37	34
3.625	35	41	41	35
4	43	50	47	43
4.375	47	53	51	47
4.625	50	54	55	50
5	54	57	59	54
5.375	63	62	64	63
5.625	64	65	66	64
6	73	69	70	73
6.375	75	74	77	75
6.625	79	77	79	79
7	84	81	84	84
7.375	90	85	89	90
7.625	89	88	93	89
8	94	93	94	94
8.375	98	98	101	98
8.625	102	107	106	102
9	106	105	109	106
10	118	117	121	118
11	129	129	131	129
12	144	141	143	144
13	152	151	157	152
14	166	167	171	166
15	179	177	179	179

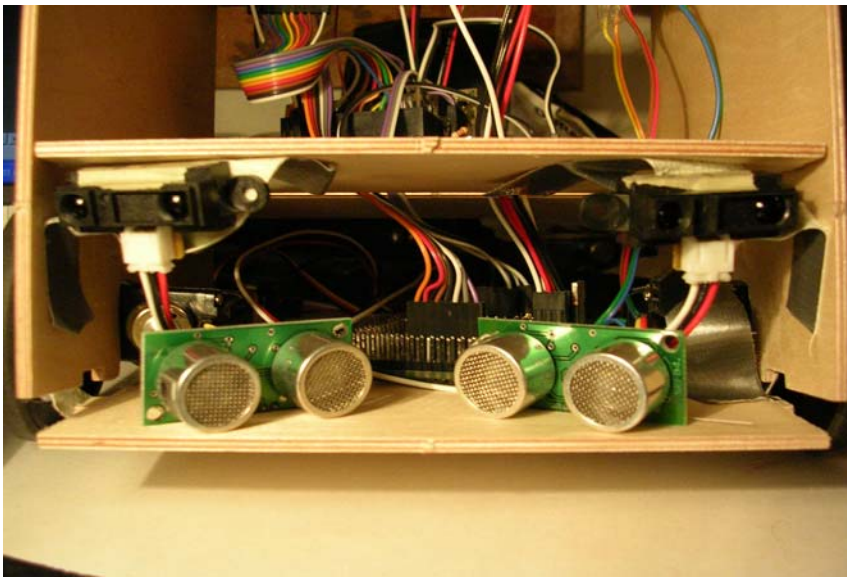
As with the GP2D12 sensors, all four are very similar and can be programmed identically.

Software Implementation

The time it takes to receive the echo is used to calculate distance. I used the following code to take measurements.

```
int timeout = 50;

 while(((PIND&0x02) == 0x02) && timeout) // port D pin1 is where echo is read
 {                                       // when echo is low reading is complete

 left_dist++;            //count # delays
 delay_10us(1);
timeout--;               //timeout makes sure loop ends
 }
```

The timeout ensures that if an error occurs and the echo pulse never goes low the robot will not be trapped in an endless loop. Another factor taken into consideration was that the accuracy of the distance readings are dependent upon on the above loop not being interrupted. To prevent this I programmed a timer interrupt every millisecond to send a ping and wait for a response using the loop above. Below I have included a picture of my platform while the positioning of the GP2D12s and SRF04s was being tested. All of the sensors had to be precisely angled for the behaviors be effective.



Bump Switches

The simplest sensor I will use is a bump switch. The purpose of the switches will be to inform the robot when the inside of its “hand” is touching an object.

Theory of Operation

Using the sonar detectors the robot will position itself a predetermined distance away from the object it wants to lift. It will then open its hand and move forward until the

bump switch is depressed. When this happens the value of the input pin connected to the bump switch will change and the robot knows to stop moving. At this point the robot can grasp and lift the object. The bump switches should not be necessary, as sonar is extremely accurate for distance calculation. They are a preventative measure and act as error detection in my behavior routines.

Software Implementation

The simple program shown here demonstrates to use of bump switches.

```
while(bump != pushed)
{
SERVO5 = SERVO_FOR1;
SERVO6 = SERVO_FOR1;   //robot slowly moves forward
}
SERVO5 = SERVO_STOP
SERVO6 = SERVO_STOP

lift_arm();
```

RF

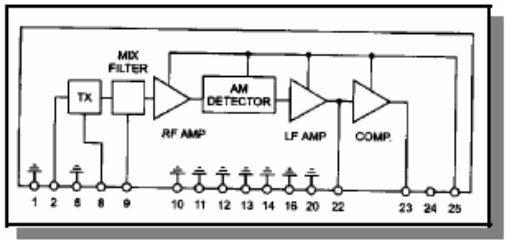
The robots will communicate using AM-RTD-315 transceivers. As canbe inferred from their name, these transceivers use amplitude modulation to communicate at a frequency of 315 MHz. The robots only need to send each other messages when they find things, but to

Theory of Operation

The AM-RTD-315 uses the TX and RX pins to send and receive serial data. To put data in serial format I used UART1 of the microcontroller and connected the input and output directly to the transceiver. When the robot has data to send it can turn off the receiver using pin25 to achieve half-duplex communication.

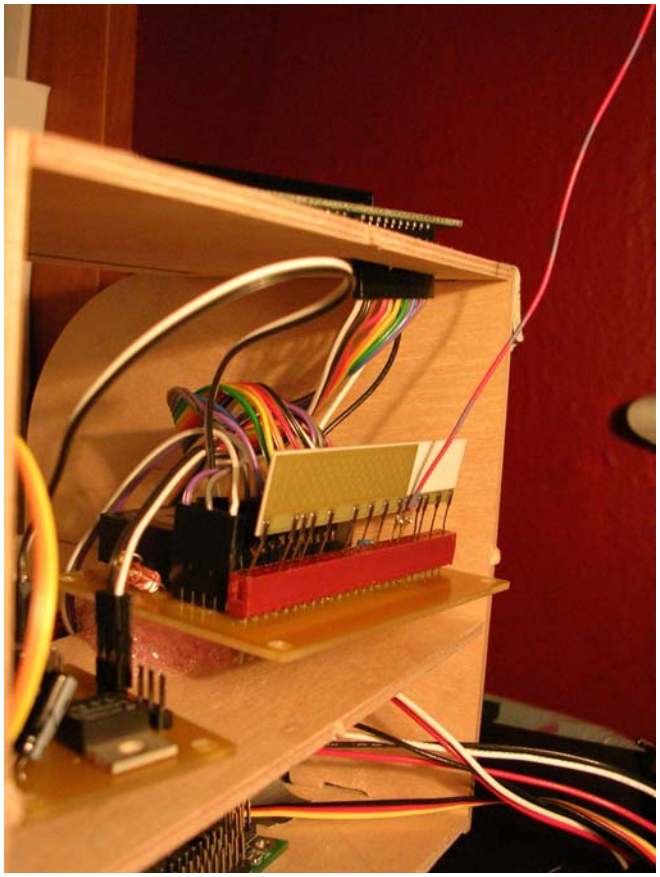
Pin Description

1) Ground
2) TX data input
   0V=Tx Off
   5V= Tx Continuous On
6) Ground
8) Tx +5V supply
9) Antenna
10) Ground
11) Ground
12) Ground
13) Ground
14) Ground
16) Ground
20) Ground
22) RX analog Out
23) RX digital Out
24) Not Used
25) RX + 5V Supply



Standard RF protocol dictates that a quarter-wavelength antenna be used, which is 8.91 in. in this case. Encoding is not necessary to send or receive data, but can ensure reliable transmission. This can be easily achieved with an encoder chip or done in software. Manchester encoding is the most simple and most widely used. I am using a parity bit as my only form of error checking to send raw data. This is because the robots will never be very far apart and messages can be sent multiple times. Without encoding the data rate is about 10kbps

The transceivers are mounted on the back of the robot with the antenna as far as possible from the batteries. This picture shows how the antenna is connected to the back of both robots.



Software Implementation

The following code shows the UART1 initializations.

```
int init_uart1(void)
{
/* enable UART1 */
  UBRR1H = (BAUD_RR >> 8) & 0xff;
  UBRR1L = BAUD_RR & 0xff;

  UCSR1B = (( 1<<RXEN) | (1<<TXEN) | (RXCIE1)); //ENABLE TX, RX AND RX
                                                  //INTERRUPT
  UCSR1C = ((UPM11) | (UPM10) | (UCSZ11) | (UCSZ10));//SET FOR 8 BIT CHAR,
                                                       // ODD PARITY
}
```

The RX interrupt was enabled to allow the robots to go about their business until they receive a message rather than use polling. When a robot needs to transmit data they can use the following routine. The receiver is turned off in the beginning and the enables again at the end so as not to interrupt transmission.

```
void USART1_TX(unsigned char DATA_TX)
{
PORTC = (PORTC & 0xFE);
          while( !(UCSR1A & (1<<UDRE)))
          ;                                  // WAIT FOR EMPTY TX BUFEFER
  UDR1 = DATA_TX;                    //PUT DATA IN REGISTER
  PORTC = (PORTC | 0x01);
}
```

One problem I have had with RF is that it must send data for about 30ms before the other end can begin receiving without errors. One solution to this problem is to send about 30ms of garbage at the beginning of every transmission. This significantly decreases the data rate, but as that is not my priority I think that this is a sufficient solution.

Conclusion

Using all of the sensors I have presented in this paper my robots will be able to avoid running into walls, search for and align themselves with various objects, know when they are holding an object, and communicate with each other. The remaining task is to integrate all of the robots abilities together into behaviors. By building on the programs I already have written, this goal can be easily accomplished.

Sources for Parts

Abacom Technologies
www.abacom-tech.com

AM-RTD-315 Transceiver modules

Mark III Robot Store
MarkIII@junjun.org

Sharp GP2D12 Infrared Sensors
3 pin JST cables

Acroname
www.acroname.com

Devantech SRF04 Ultrasonic Sensors

Hobby Shack
www.shopatron.com

Hitec HS-422 Standard Deluxe Servos

Servo City
www.servocity.com

Hitec HS-81MG Servo