EE-318 Electronic Design Lab (EDL) Project Report
Remote Controlled Smart Mote

Group no. 2

Group Members:

Neel Mehta - 07d07001
neelmehta89@gmail.com
Prateek Karkare - 07d07002
prateek.karkare@gmail.com

Faculty Members:

Prof. Maryam Shojaei Baghini
Prof. Dipankar

Background/Motivation

The project is based on the principle of swarm robotics. In swarm robotics, we have lots of motes (small robots) that communicate which other using artificial intelligence to complete a task. These tasks cannot be done by them individually, but together as a group they can complete a task successfully. We have simplified the problem to a light sensing device using 2 motes. This robot describes the basic principal behind the swarm intelligence and finds its applications in many areas like defense, rescue operations, explorations etc.

Customer Specifications

- There are two moving robots which are equipped to find the light source and approach it.
- On the condition that anyone of them reaches the light source it should indicate the other one about its location.
- The other one should be able to approach the location of the 1st one.

High Level description

The light sensing part

This consists of a small Phototransistor which is very sensitive to visible light which takes the input and feed it to the microcontroller which upon processing tells the robot to follow a definite path.

The guiding signal

The robot which has reached the light source first will have to tell the other robot about its status and should guide it to reach at the same position. Obviously this task has to be done using some signal which has to be more powerful than the light source. We chose IR for this purpose since it can be detected from a longer range and also it requires less power.

The brain and motor controller

All the processing is done by the microcontroller Atmega 16, the robot first rotates and searches for the light source the inputs from the phototransistor are sampled at a high rate and then it estimates the angle at which the light source must be and follows that path. Obviously there will be some error in the angle estimation there are two main ways to handle this:

We can use a continuous feedback mechanism which continuously takes input from the sensor and keeps correcting its position

Or, we can keep correcting the error after moving a particular distance and then again searching for the direction of the light source.

Given the limitations of the sensor and the complexity of the code in the microcontroller we opted for the second approach.

The motor controller we are using is L298 which is a full H-Bridge motor driver with the advantage of being able to handle a large current and sensing the current required by the motor.

Technical Specifications

- The mote consists of a light sensor (Phototransistor – L14G2) to trace the position of the light source.
    - The sensor is very sensitive to visible light and has a quite fast response time (response 7µs on time, 8µs off time).
    - Since it is a phototransistor the base current bias gives much flexibility to calibrate it to ambient conditions.
    - It has a quite sharp sense angle (+10 deg to -10 deg for half sensitivity).

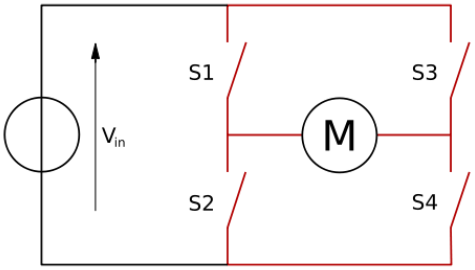
Basic principle of operation:-
Phototransistors consist of a photodiode with internal gain. A phototransistor is in essence nothing more than a bipolar transistor that is encased in a transparent case so that light can reach the *base-collector junction*. The electrons that are generated by photons in the base-collector junction are injected into the base, and this photodiode current is amplified by the transistor's current gain β (or h_{fe}).

L14G2: The Phototransistor we are using

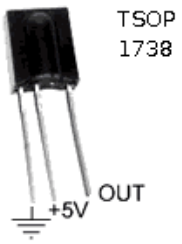
- The analog input is then amplified and fed into a microcontroller (ATmega16) for processing.
    - ATmega16 is a powerful microcontroller with advanced RISC architecture.
    - It has quite a large nonvolatile program and data memories.
    - Two 8 bit timers and one 16 bit timer and 4 PWM channels.
    - One 8 channel 10 bit ADC with programmable gain and inbuilt analog comparator.

- The whole processing and commands are given by the microcontroller to the motor driver (L298)
    - L298 is a full H-Bridge motor driver which can drive two motors.
    - Peak output current capacity per channel is 600mA.
    - A current sense pin which is quite helpful to control the maximum current to the load (motor).

The basic structure of the H-Bridge circuit is as shown below.



- The microcontroller will drive two DC motors (geared, RPM = 70, Voltage = 5V) using L298.
- For the beacon signal we will be using an IR LED. The IR LED will start glowing as soon as the mote reaches the light source.
- For the receiver TSOP 1738 is used. The output of the TSOP is fed into the uC.
  - Works on a carrier frequency of 38kHz.
  - Photodetector and preamplifier in one package.
  - Built in filter for PCM frequency.
  - Low power consumption.
  - High immunity against ambient light.
  - Provides very good range for small bursts.



Implementation

We are using phototransistors to detect the direction of light source. The mote rotates and samples for the light source using the phototransistor. The analog output signal of the phototransistor is fed to the microcontroller.

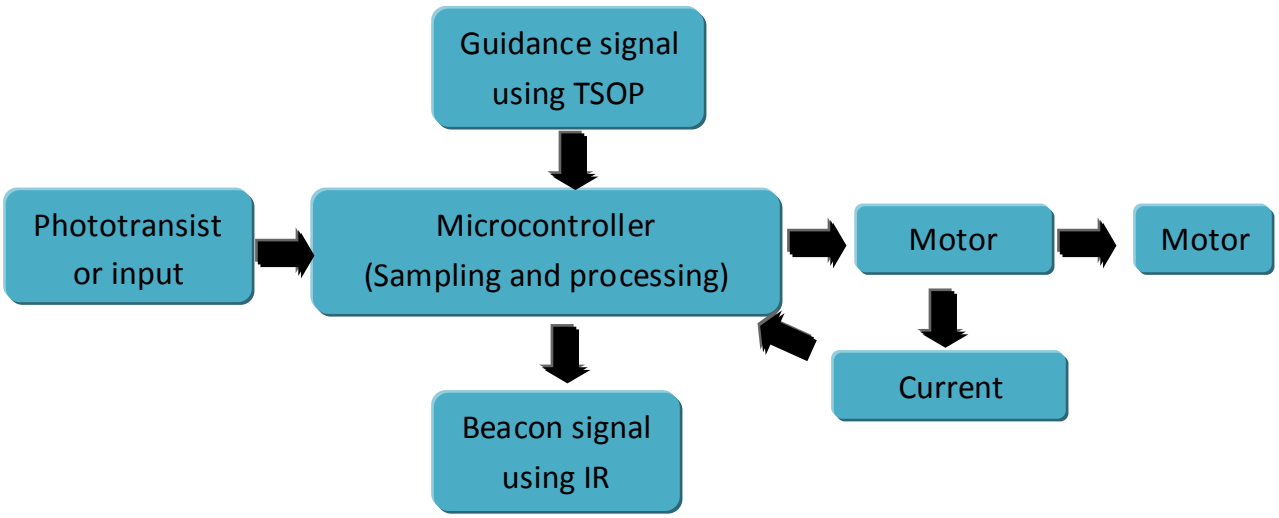
The microcontroller uses the ADC to find out the digital value. Then the samples are averaged out over an interval of 10 degrees, and the minimum is used to detect the direction of light source. It then directs the motor driver circuit to rotate the mote in the direction of the light source and then move linearly in the direction of light.

After moving for about 2-3 seconds, it again repeats the above procedure – sampling, averaging, finding peak, rotation, and linear motion to correct for any error during motion of the car due to varied rpm (small, but if any) of the two dc motors. This process goes on till the mote reaches the destination i.e. the light source. As soon as the mote reaches the light source and collides with it, the surge in current in the motor is used as a stop signal and

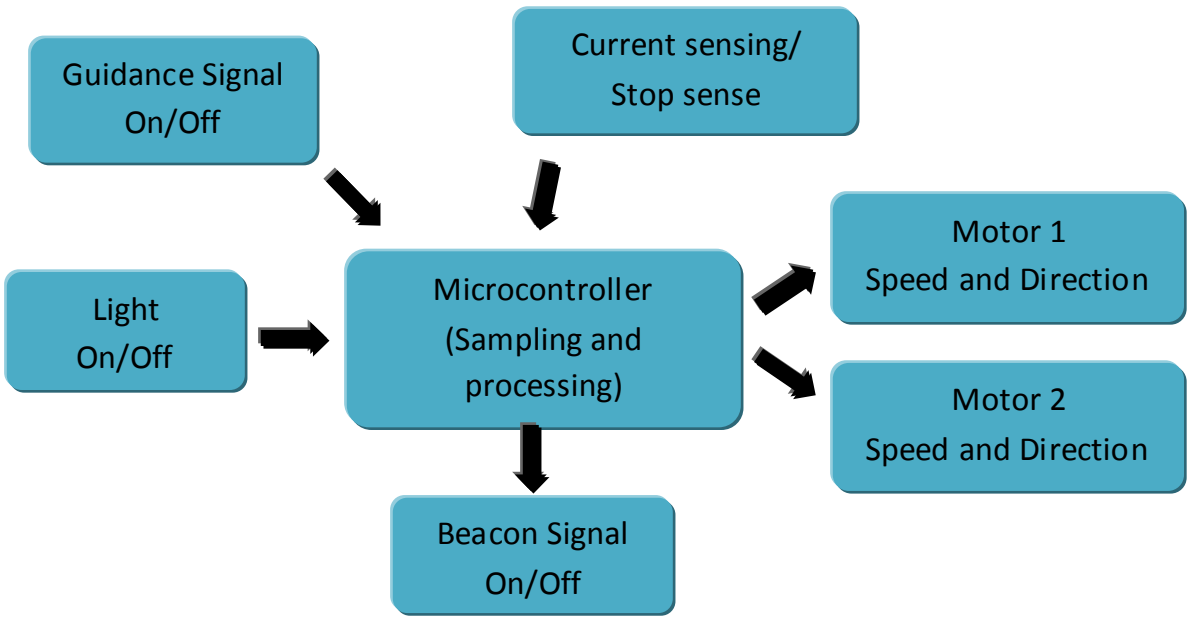
provided as a feedback to the microcontroller. The microcontroller thereby stops further movement of the mote, as our mote has reached the destination light source.

After the mote has reached the light source, it needs to signal the other mote that it has reached the destination. It turns on the IR LED to achieve this. The other mote (exactly identical to the first mote) at the time of sampling for light also searches for any IR light of frequency 38kHz. So there is an alternate sampling of light (light source) and IR light. If the mote detects any IR light, it means that the other mote has reached the destination. So, after completion of the rotation, it moves in the direction of the first mote, after turning in the direction of light source. It remembers the direction of IR source just like the light source, during the rotation.

Block Diagram



Software Block Diagram



Software flow

/**
Global Definitions of variables
Defined the global and shared variables required by different functions.
**/

/**
Initialise the ADC
Set the ADMUX and ADCSRA registers in the required mode to enable ADC in the Atmega 16
**/
void adc_init(void)

/**
Sum the sample
Take the higher order 8 bits from the ADC, i.e. ADCH. Add 100 values/samples for each 10 degrees
**/
void sum_add()

/**
Find the minima in the array
Find the minimum value in the array. The angle corresponding to minimum value will point to the direction of light source.
**/
unsigned int find_min()

/**
Move the motor linearly
Set the direction for motors. Set the PWM for speed control. Run the motor linearly for 5 seconds (may be varied, depending on need), and keep on checking for stop sense. If stop sense is observed, then stop the motors and enter the IR stage. Keep on outputting the modulated 38kHz wave as long as the reset switch is not pressed. If no IR found, stop generating PWM, after the 5 seconds time.
**/
void linear_motor()

/**

Rotate the motor by a fixed angle
Set the direction of rotation for motors, and set the PWM. There is no delay here, as we use this function to sample and rotate and the machine, and rotation and sampling occurs side by side.
**********/
void rotate_motor_fixed()

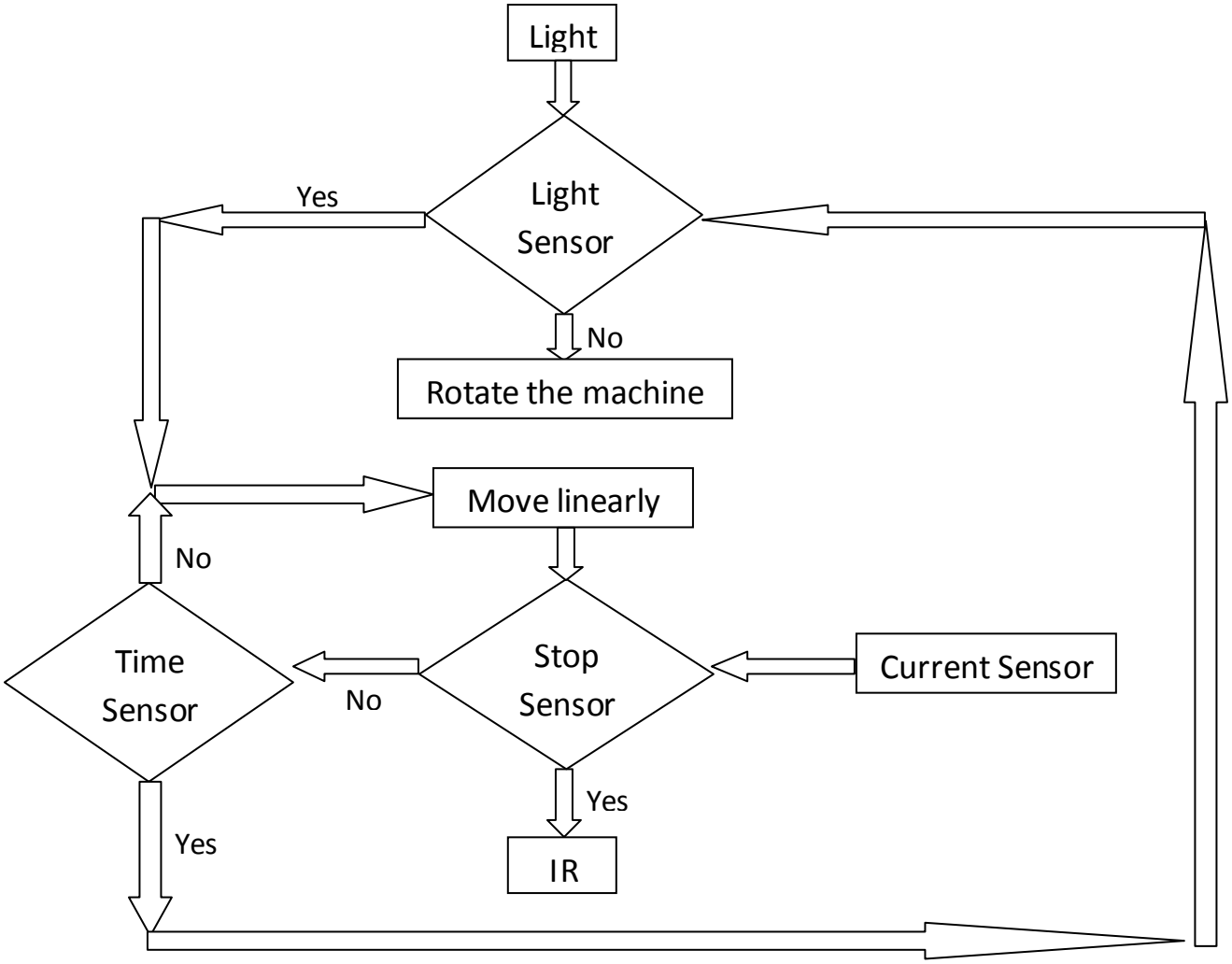
/**********
Rotate the motor by the required angle
Set the direction of rotation of motors. Set the PWM. Set a delay proportional to the required rotation angle. Stop generating the PWM.
**********/
void rotate_motor(unsigned int rot_angle)

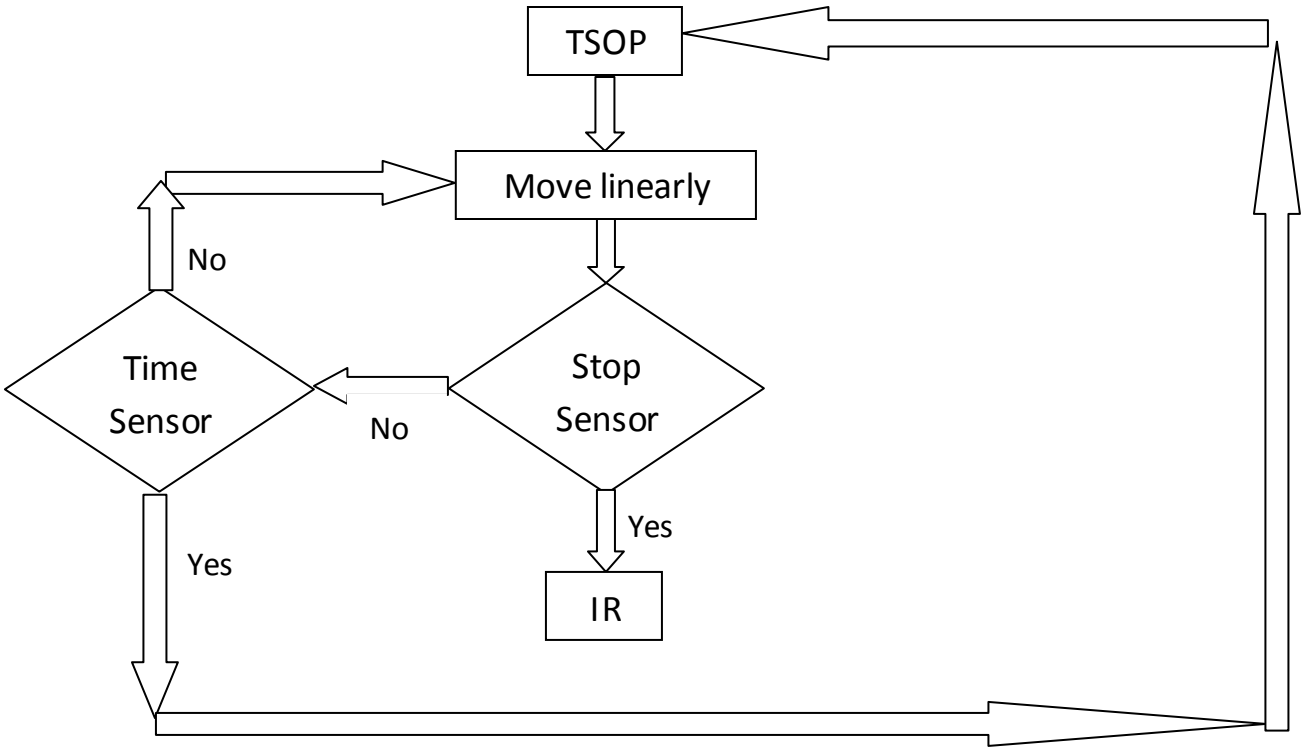
/**********
ADC function
Initialise the ADC to start converting analog to digital taking input from PinA0. Wait till conversion completes. Use sum_add() to add this digital value to an array. Check for TSOP input, and see if any IR light has been detected or not. If detected enter into TSOP mode. Repeat this procedure till 100 samples are taken per 10 degrees, for complete 360 degrees. Then stop the mote, find the minima, rotate the mote to align it in the direction of minima, and then move linearly in that direction.
**********/
void adc_read()

/**********
Initialise the timer
Initialise the timer by setting the TCCR1A and TCCR1B registers.
**********/
void timer1_init (void)

/**********
Main
Set the data direction registers. Call the initialization function of the ADC and the timer, and then call the adc_read() function indefinitely.
**********/
int main (void)

Software Flowchart



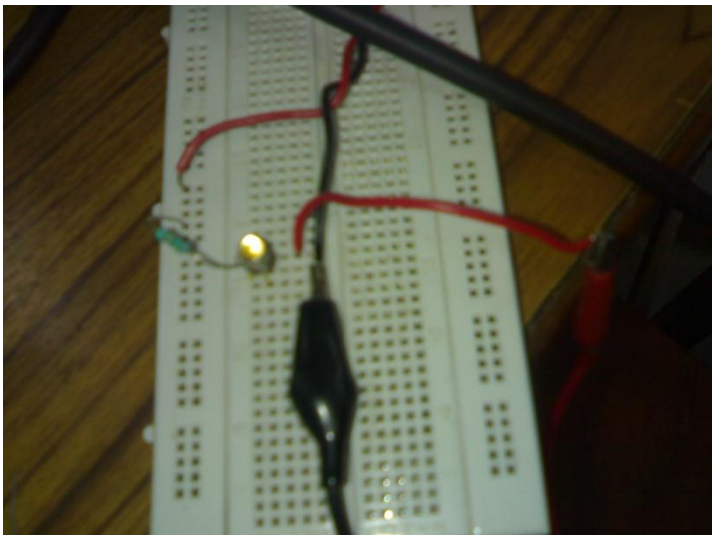


Hardware circuit

Sensor circuit

We will be using a light sensor to detect the light source which the mote has to locate, for that we will be using a phototransistor (L14G2) which has quite fast a response time (7 µs on time and 8 µs off time) and 20˚ (±10˚) which will serve our purpose quite well.

For the second mote which will be directed by the first mote we have used IR led and sensor as guide for which we have used TSOP. The IR LED is on for a period of 0.5 ms and off for a period of 5 ms. So, during the 0.5 ms, 19 cycles pass of modulated light wave pass through. The modulated light is used because using IR LED in this manner increases the sensitivity of the TSOP, and thereby its range. In this manner it is able to detect IR light for ranges upto 1.5 m. This range can be further increased by increasing the IR light intensity and thereby increasing its transmitting power. We do not want the power requirement to increase a lot and hence did not increase the transmitting power.



Microcontroller circuit

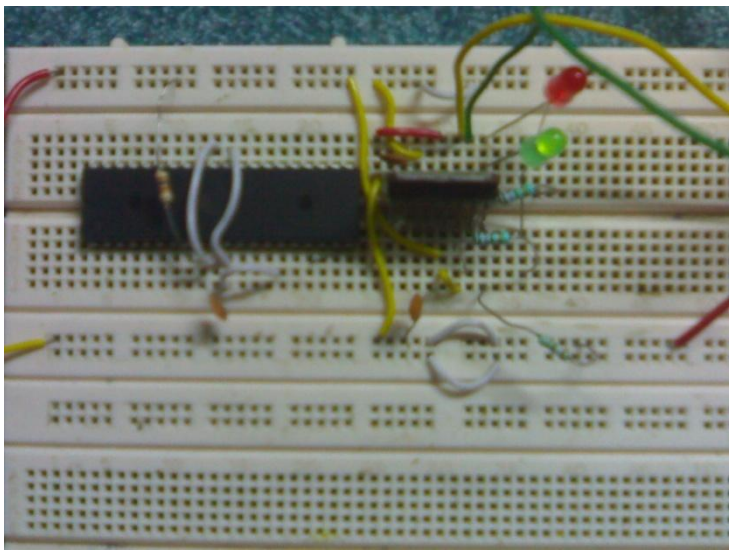
Basic circuit for the microcontroller with interfaces with sensor, driver etc. The interface with the phototransistor is via ADC of the uC. The interfacing with the TSOP is using a simple input pin. The motors are controlled through L298 using PWM, enable from the uC and stop control as the feedback to the uC.

Motor driver circuit

We have used L298 motor driver to drive out two DC motors L298 provides max. 600mA current which is quite above our requirement of 200mA current also it has a current sensor pin which can solve our problem of sensing the current beyond a threshold point and stopping the device.

L298 is a full H-Bridge driver which has four transistors connected in an “H” formation thus the name H- Bridge. It has input pins for the PWM waveform which drives the transistors

inside the H-Bridge and supplies the current to the motor current sensing pins are nothing but the emitters of the transistors which can be connected to the ground directly or we can connect a small resistance in series to sense the current flowing through them and hence through the motor



Experimentation with L14G2

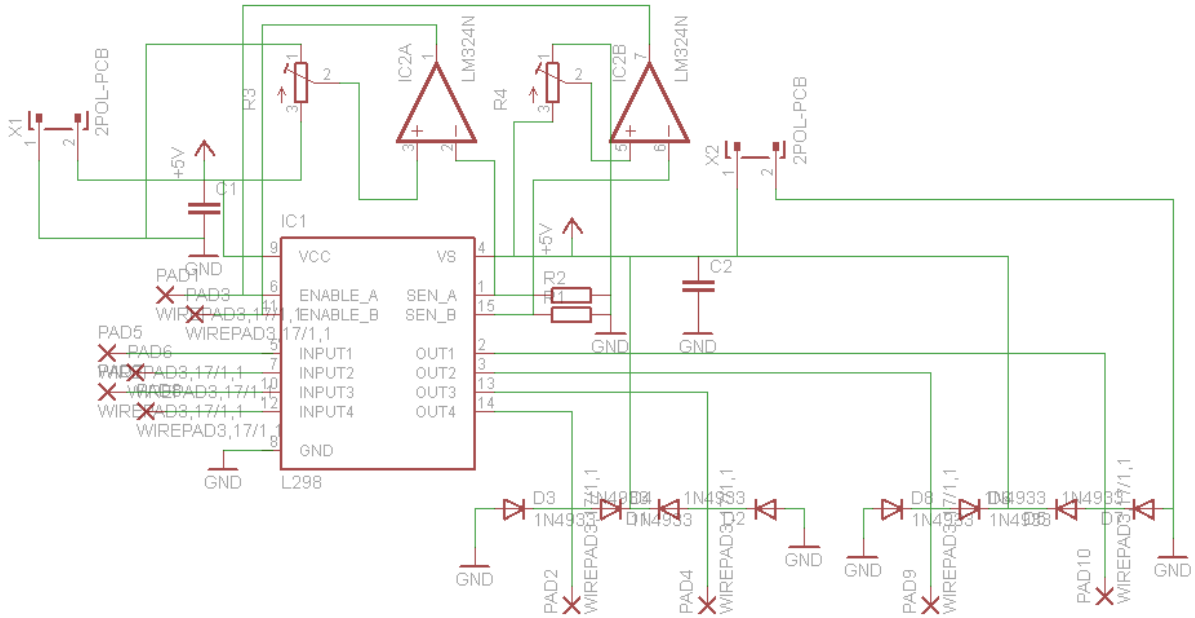
Initial part of the experiment required a good sensor with high range. We looked through and experimented with many light sensors like photodiodes, LDR etc. Our requirement was a sensor which should have a small sense angle so that we can determine the direction of the source with greater precision, one feature of this sensor which later proved to be an unwanted thing to us was this transistor was highly sensitive and it even detected the variations in the light source due to voltage changes. For instance the waveform which we observed on CRO when the phototransistor was exposed to ambient light was a sin wave of freq. 100Hz which was corresponding to the fluorescent tubes. We solved this problem inside the software instead of using filters of ambient light cutoff.

Experiments with TSOP1738

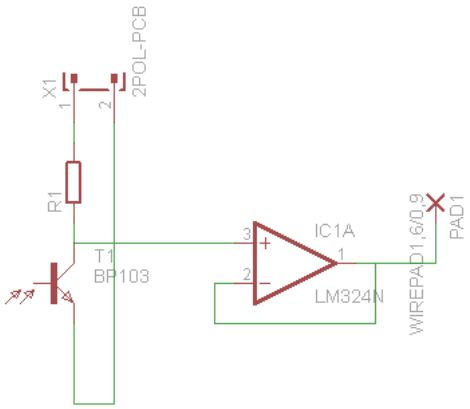
TSOP1738 is an infrared detector with built in preamplifier. It gives a low output when it detects an IR signal. The device is designed to work on frequencies of 38 KHz square wave so that it can avoid ambient noise signals. With the 38 KHz wave the range of the transmitter is very less (about 8-10 cms) which is far from our purpose as we required about 150 cms of range. After carefully studying different circuit implementations in the TV remote control and the datasheet of TSOP1738 we fed a modulated burst signal waveform to this sensor the modulation scheme was:

Carrier frequency: 38 KHz square wave
This wave was transmitted for 1 ms and then was switched off for 5 ms so in effect we superimposed two waveforms, one 38 KHz square wave and other 0.166 KHz wave with 16.6% duty cycle
The range increased by about 100 cms and we finally got the desired range
One more advantage of this modulation was that we were able to push more current through the IR transmitter diode as this was pushed for a very short duration the diode was able to withstand it and thus helped in increasing the range even more

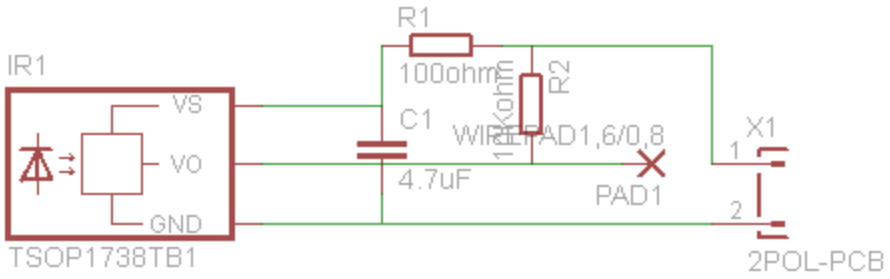
Schematics



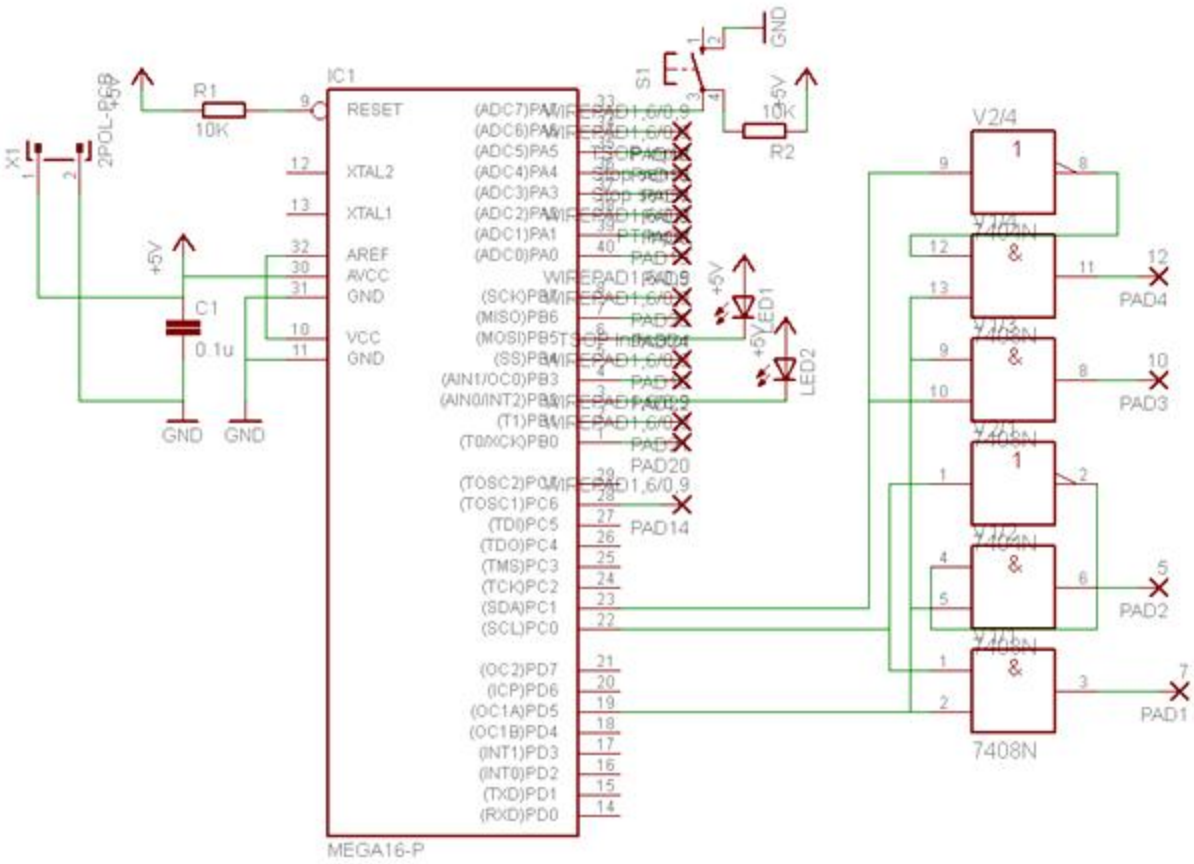
Driver and current sensing circuit



Phototransistor sensor



TSOP sensor circuit



The Brain: Microcontroller and logics

Work Flow

Our project was finalized on 1st February 2010. We worked the next 2 weeks on:

1. Photo-sensors, trying to experiment with photodiode, LDR, phototransistor, etc. We finally found out that the phototransistor suited our purpose best because of its high sensitivity and range. Also, we could control the sensitivity by controlling the base-emitter bias.
2. IR LEDs, TSOP, IR receivers, etc. We could achieve proper reception for small range only. We were unable to get a large range at that point of time.
3. Use the phototransistor collector voltage as an input to the ADC of the microcontroller, and convert it into the digital value.

After midsem, we divided the working among ourselves. Neel started working mainly on the software part, and Prateek mainly on the electronics part. We coordinated and helped each other out as and when required. In the month of March, the following was completed:

1. Build a car as per our need to be light weight and small.
2. Work on the motor driver circuit. Use L298 as the driver IC, and interface them with the DC motors. We found out that the 12V/100rpm motors that we had consumed a lot of current, thereby a lot of power. So, we had to cut on the power requirement, and hence we needed some low power DC motors. So we got the 3V/70rpm motors.
3. Also, wrote the code for the PWM generation and used them with the new DC motors. Also used the same PWM to run both the motors using additional digital ICs.
4. Wrote the code of the complete analysis of the light samples – ADC, average and find minima to get the direction. Then rotate the mote in the direction of light source, and then move linearly in the direction of the light source. This required proper generation of PWM and the direction signals for interfacing them with the L298.
5. Also, incorporated the stop sense/current sensing, and provided feedback to the DC motors to stop them.

In the month of April, we worked on:

1. Proper calibration of the car to make proper 360 degrees rotation, by changing the PWM to reduce speed and decrease the number of light samples taken to find the average value to help in predicting the direction of light source.
2. Change the stop sense feedback to the L298 via uC, instead of direct feedback, because there was a problem due to current sourcing/sinking. Also, we needed the uC to know that the mote had reached the destination, and to glow the IR LED.
3. Worked on the IR LED and TSOP, working out many different transmitter circuits, and different modulated light. Finally we achieved success by using a BJT for current gain, so as to provide high current to the IR led. Also, used a modulated wave as described before to generate the modulated wave. This provided us with a modulated light wave and we were able to achieve range of around 1 meter. We found that the 555 timer IC was unable to source enough current, and that is when we used the BJT. But still there was quite a large drop in the 555 output peak voltage and thereby affected the range that we could achieve adversely. So, we used buffer for this purpose. But all these additional ICs led to a considerable increase in the current and thereby the power required. So, to cut on the power we generated the modulated wave using the uC. Now, we could achieve ranges of the order of 1.2-1.3 meters.
4. We had to measure the TSOP voltage, and find any detection of the IR LED. We tried out various circuits using JK flipflops but could not achieve success. Hence, we resorted to using the uC as a device to sense any pulse in the TSOP output. So, we

provided the output of TSOP as a feedback, and then align the car in that direction, and move the car in that direction. So, we changed the code accordingly to incorporate all this.

List of components required

Phototransistor (L14G2)
Logic IC's
Microcontroller (ATmega16)
Motor driver (L298)
Buffer (using LM324 – opamp)
Comparator *(LM324)*
TSOP
IR LED
DC Motor (Geared)(70 RPM)

Problems faced

Major problems were faced in choosing appropriate light sensor and calibrating it to the ambient conditions L14G2 is very much sensitive but has quite adjustable sensitivity which can be achieved by biasing the base with appropriate base voltage.

Another problem was to develop a long range IR emitter similar to the one's which are used in TV remote controls , since we found out by experimenting that the TSOP works fine with TV remote control but does not work well with the standard IR emitter circuit. We could solve this problem using modulated wave for transmitter, instead of constant 38kHz wave.

Another problem was in calibration of the time intervals for proper rotation of 360 degrees of the machine. Also, it was difficult to align the sensors for proper reception. We had to pay this price of extreme sensitivity to angle to gain long ranges.

Scope of development

Swarm robotics is a burning research topic these days to imitate the behavior of natural swarms and swarm intelligence algorithms are under rapid development, our basic idea can be extended to many small motes to make a swarm which can be further extended to 3D swarms (flying motes) these can be implemented to many areas like expedition projects, scientific researches and data gathering in remote and dangerous areas, military applications, rescue operations etc. We wish to extend this further to such applications by incorporating more sensors to detect the environment and developing intelligent swarm algorithms for different applications

Software Code

```
#include <avr/io.h>
#include <util/delay.h>

long array[36];
unsigned int num = 0, f1 = 0, angle=0;

void adc_init(void)
{
	ADMUX = _BV(REFS0) | _BV(ADLAR);
	ADCSRA = _BV(ADEN) |_BV(ADPS2)|_BV(ADPS1);
	/* Use AVcc as reference, Left adjust the result
	Enable the ADC, use prescaler = 64 for ADC clock */
}

void sum_add()
{
	array[angle] = array[angle] + (long)ADCH;
	num++;
	if (num > 100)
		f1 = 1;
}

unsigned int find_min()
{
	unsigned int i, min_angle = 0;
	for (i=0; i<=35; i++)
		if (array[i]<array[min_angle])
			min_angle=i;
	return min_angle;
}

void linear_motor()
{
	PORTC = 0x03; // 11 set right and left motor to move in fwd direction
	OCR1A = 0xFF; //5V in hexadecimal
	int i;
	for (i=0; i<10; i++)
	{
		if (((PINA & 0x04) == 0) | ((PINA & 0x08) == 0))
		{
			while ((PINA & 0x80) == 0)
			{
				PORTB = 0xF0;
				for (i=0; i<19; i++)
				{
					PORTC = PORTC | 0b01000000;
```

```
                                        _delay_us(11);
                                        PORTB = 0xF0;
                                        PORTC = PORTC & 0b10111111;
                                        _delay_us(6);
                                }
                                _delay_ms(5);
                        }
                        PORTB = 0xFF;
                        break;
                }
                _delay_ms(500);
        }
        //_delay_ms (20000);
        OCR1A = 0x00;
        _delay_ms(2000);
}

void rotate_motor_fixed ()
{
        PORTC = 0x02; // 10 set right and left motor to move in opp direction - rotate car
anticlockwise
        OCR1A = 0xCC;
//      _delay_ms (20);
}

void rotate_motor (unsigned int rot_angle)
{
        PORTC = 0x02; // 10 set right and left motor to move in opp direction - rotate car
anticlockwise
        OCR1A = 0xCC; //2V in hexadecimal
        _delay_ms (100*rot_angle);
        OCR1A = 0x00;
}

void adc_read (unsigned char channel) // valid options for channel : 0 to 31. See datasheet
{
        ADMUX = (ADMUX&0xe0) + channel;
        // Set channel bits in ADMUX without affecting other bits
        ADCSRA |= _BV(ADSC); // Start conversion
        while((ADCSRA & _BV(ADSC)) != 0) {};
        // Do nothing until conversion is done
        //return(ADCH); // Return upper 8 bits

        sum_add();

        if (angle!=0)
        if ((PINA & 0x10) == 0)
```

```
	{
		PORTB = 0x0F;
		while ((PINA & 0x80) == 0);
		PORTB = 0xFF;
	}

	if (f1)
	{
		array[angle] = array[angle]/100;
		angle++;
		num = 0;
		f1 = 0;
		if (angle < 36)
			rotate_motor_fixed();
		else
		{
			OCR1A=0x00;
			_delay_ms(2000);
			angle = 0;
			unsigned int min_angle = find_min();
			rotate_motor(min_angle);
			_delay_ms(2000);
			linear_motor();
		}
	}
}

void timer1_init(void)
{
	TCCR1A = _BV(WGM10) | _BV(COM1A1); // Fast 8-bit non-inverting
	TCCR1B = _BV(WGM12) | _BV(CS11); // PWM with prescaler = 8
}

int main(void)
{
	DDRC |= _BV(PC0);    //direction for motor1 - left
	DDRC |= _BV(PC1);    //direction for motor2 - right
	DDRC |= _BV(PC6);
	DDRD |= _BV(PD5); // Necessary to set DDR value for PD5

	timer1_init();
	adc_init();

	DDRB = 0xFF;
	PORTB = 0xFF;

	//	PORTC = PORTC & 0b11111011;
```

```
	while(1)
	{
		adc_read(0); // Set duty cycle

	//	PORTC = PORTC | 0b01000000;
	//	_delay_ms(5);
	//	PORTC = PORTC & 0b10111111;
	//	_delay_ms(1);
	}
	return 0;
}
```