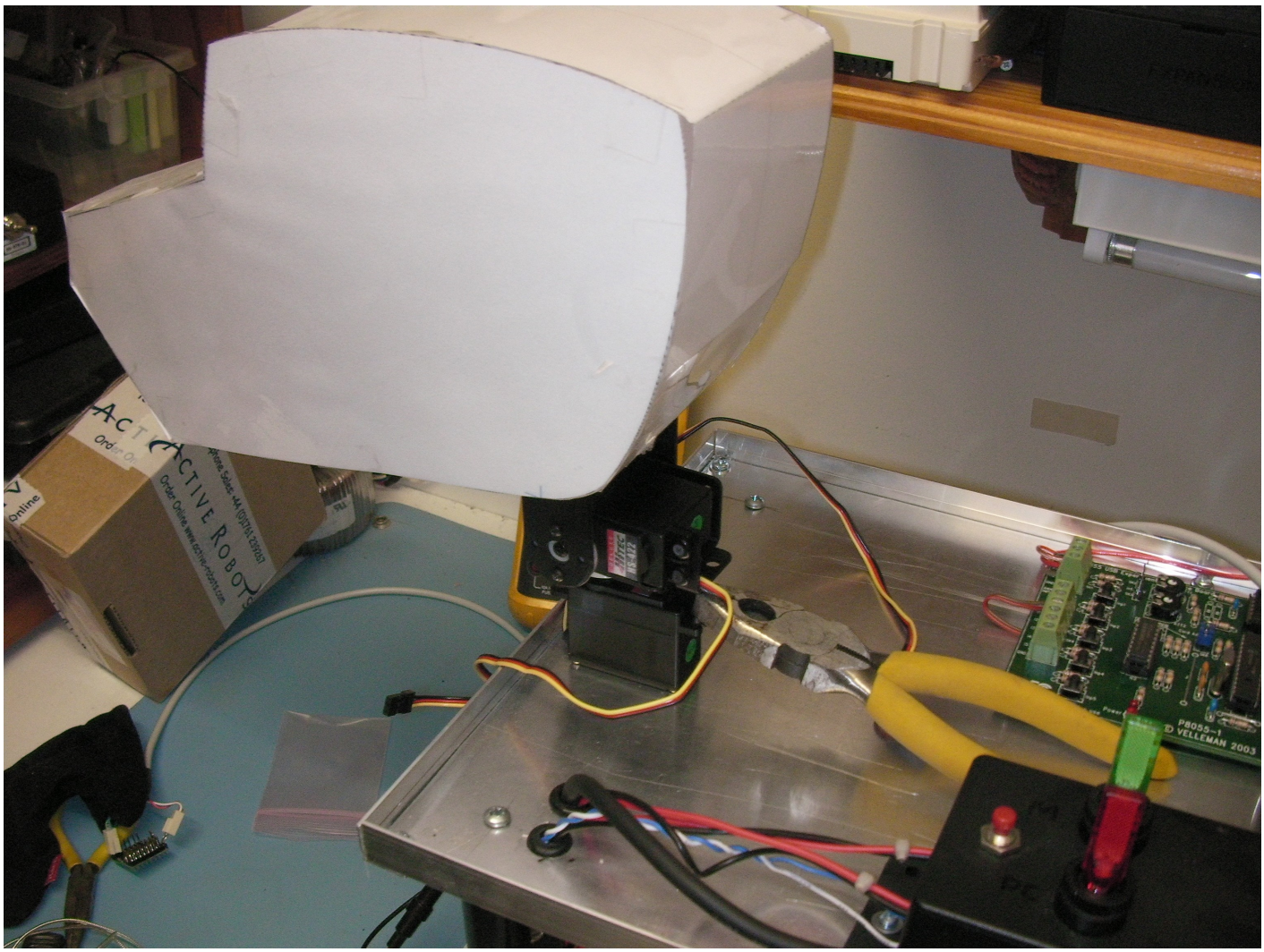
K9 AUTONOMOUS ROBOT DOG



CONCEPT / INTRODUCTION

To produce a computer controlled robot using high level programming, which can move around a room or rooms in a house and respond to certain inputs. This will include obstacle avoidance and hopefully recognition of living things via heat and sound detection. The computer will process the inputs of the various sensors as well as driving the motors via various input and output circuitry. This project started in January 2009 and I'm not sure when I will declare it finished. The Robot is my own design, although I did get various ideas from the internet. Some of the design is far from perfect, but maybe this document will help you in making your own robot. www.societyofrobots.com was particularly useful.

Contents

Initial Spec / Requirements

Power
I obtained a number of good second hand 10 cell 12V NIMH batteries for free, so I will be using a couple of these to power the robot.
Motherboard : 10 Cell 12V NIMH Battery Powered
Motors : 10 Cell 12V NIMH Battery Powered.

CPU
Computer Controlled using small low power ITX Motherboard and CPU.

Motors
Front wheel control using two DC motors.
Move at a variable speed from a craw, up to walking pace.
Use differential drive to move the robot.

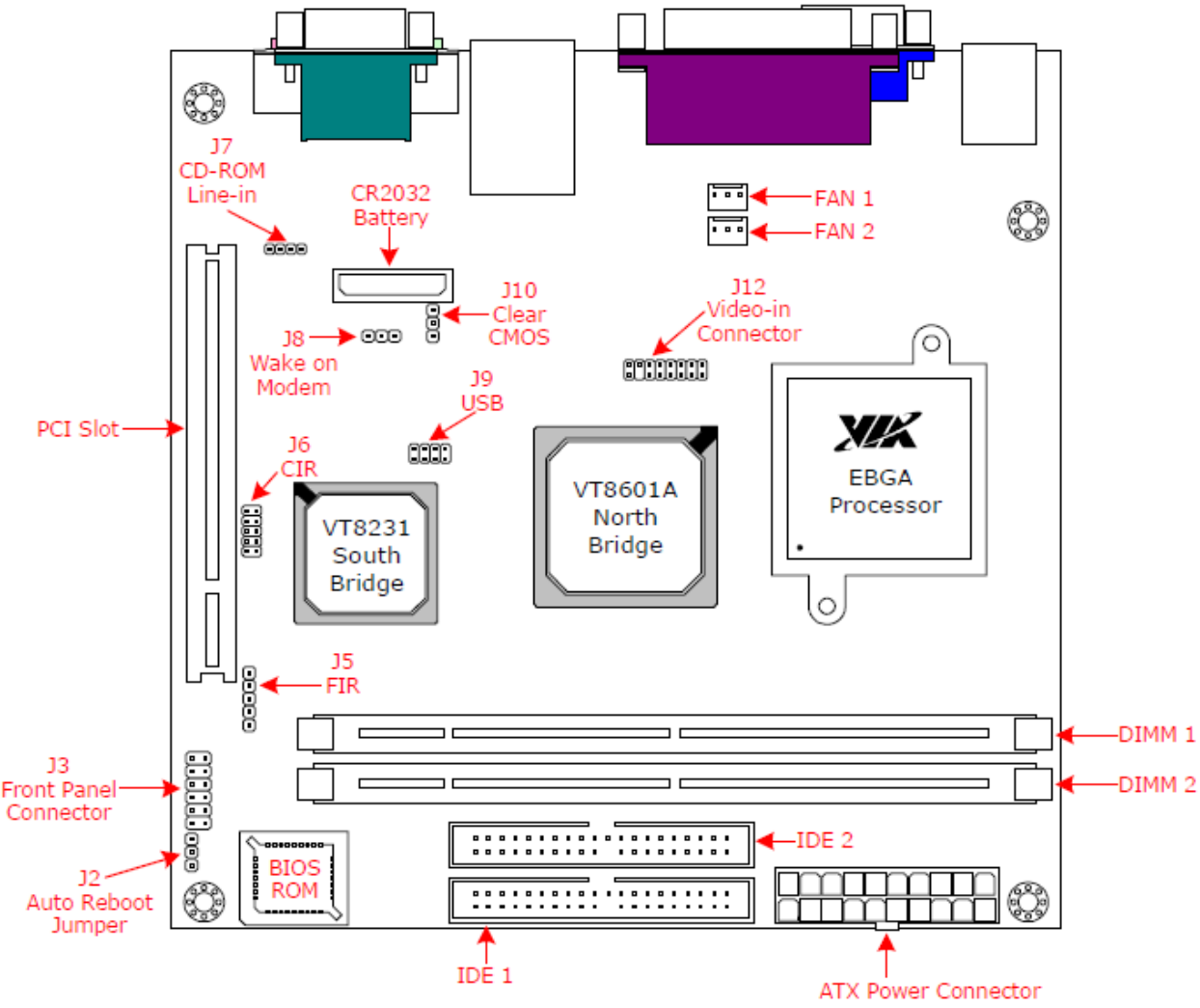
Sensors:
2 IR Range finders 4-30cm & 20-150cm
PIR thermal detector
Microphone input to motherboard.

Interfacing
K8055 USB interface board
- controls all motors
- receives feedback from encoders
- monitors battery voltages

Parallel Port
- Receives input from various analogue sensors

COMPUTER

The robot will be controlled by it's own computer, attached to the chassis.
I chose a Mini-ITX MOTHERBOARD including VIA C3 EDEN EBGA Processor running at 533MhZ for it's low power consumption.



Instead of using a Hard Drive, my set-up uses a 1GB Compact Flash card as the boot device. A compact flash card has the following advantages: Lower Power consumption , Silent, No moving parts hence less likely to be damaged. I have installed a cut down version of Windows 98SE. The boot time for the computer is approximately 30 Seconds. Control of the motors and other sensors, as well as gathering information will be handled by various interfaces to the parallel, serial and USB ports.

Power Measurements

A 12V power supply was connected to the PICO PSU and the following currents measured:

Motherboard Power Requirements

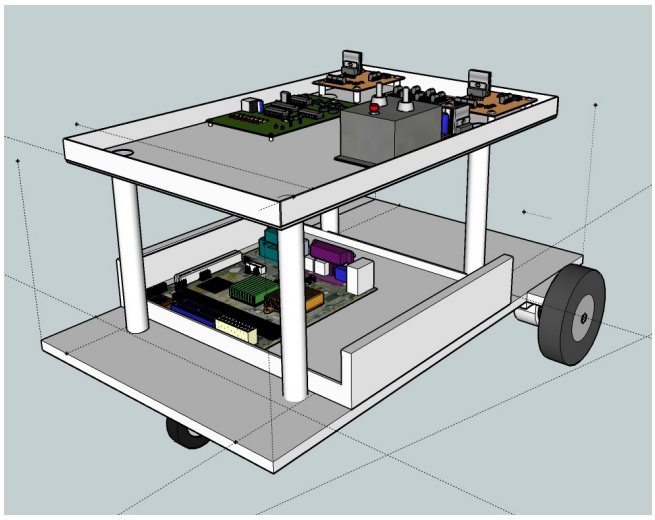
	Current	**Power (W)**
Motherboard Off	57mA	0.7
Motherboard On – after boot up, with Compact Flash Drive *No disk access*	560mA	6.7
Motherboard On – after boot up, with Compact Flash Drive. *Peak with disk access*	970mA	11.6

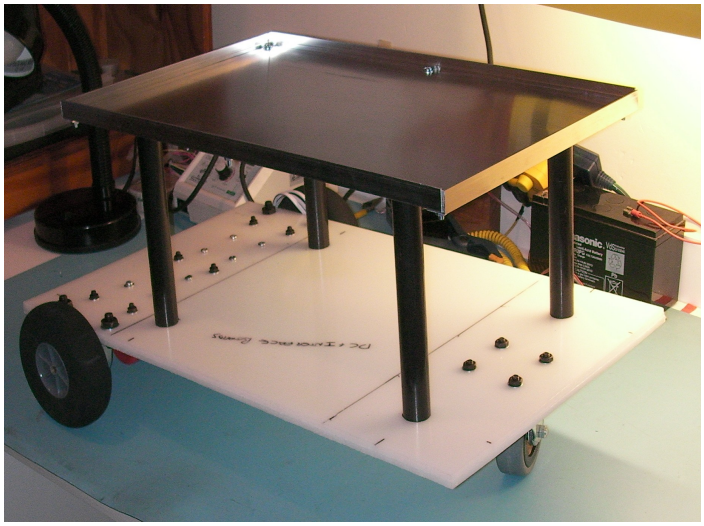
CHASSIS / BODY

The design of the chassis was done using Google SketchUp 7. The base of the chassis is made from HDPE (High Density Poly Ethylene). I chose this as it came recommended by the society of robots (www.societyofrobots.com)

HDPE has the advantage of being relatively cheap, lightweight and strong. It is also very easy to work with.

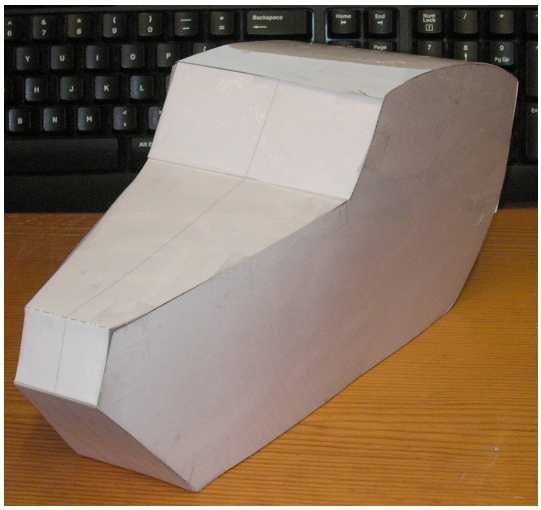
The chassis consists of a lower shelf on which the two drive motors are mounted. The motherboard is also mounted on this shelf. An upper shelf is used to mount some more electronics.





As this is my first robot, I have experimented with various materials. The top shelf is made from 2mm aluminium sheet and supported on four polycarbonate pillars.

Model of the head. The head will be mounted on the chassis in such a way that it can pan and tilt with the aid of two stepper motors. Three sensors will be mounted on the head : 2x IR range finders and a PIR thermal detector.



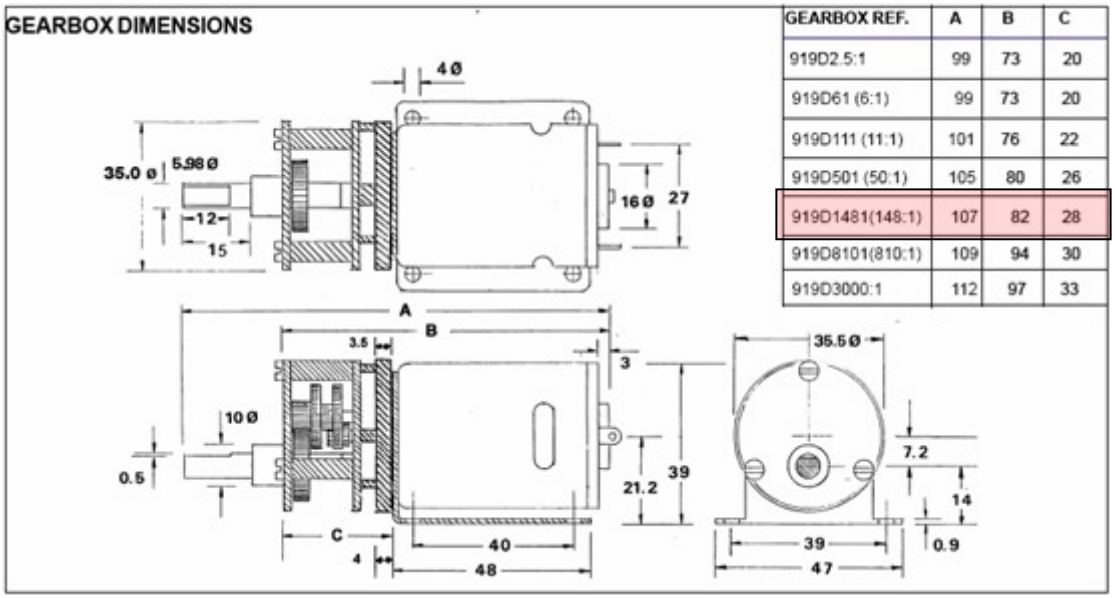
Lower Chassis Frame picture here

DC MOTORS

Two DC motors provide the drive for the robot. I've chosen the model below, which should give plenty of torque and a max rpm of probably about 110 at 12V. This motor is designed for heavy-duty industrial and model applications, so hopefully will be suitable.

Motor specification:

Voltage	rpm	Torque (g/cm)	Stall Torque (g/cm)
4.5	40	---	---
6	53	17493	99160
9	80	---	---
12	**106**	**22851**	**148000**
15	132	---	---
Loading - 3 amps continuous, 5 amps peak only			

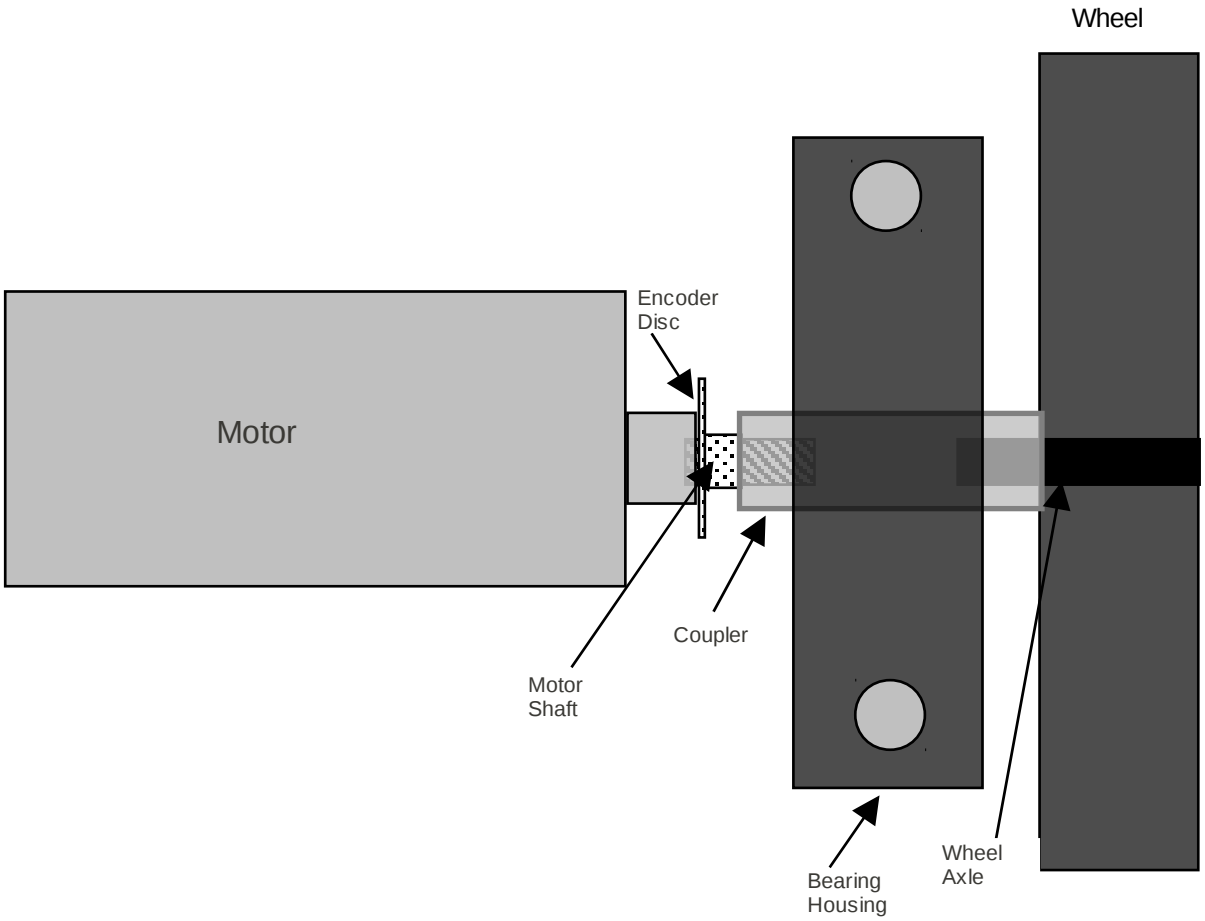


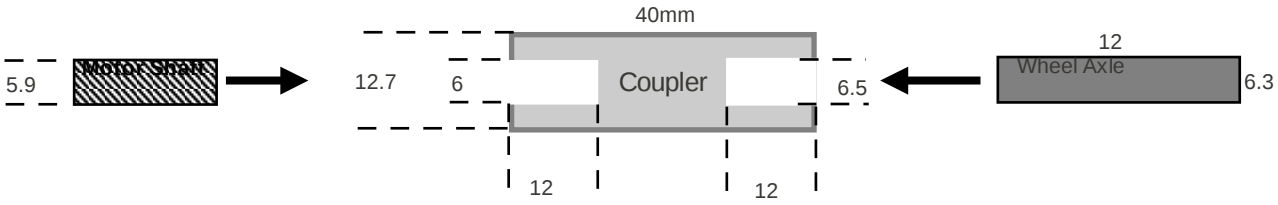
GEARBOX REF.	A	B	C
919D2.5:1	99	73	20
919D61 (6:1)	99	73	20
919D111 (11:1)	101	76	22
919D501 (50:1)	105	80	26
919D1481(148:1)	107	82	28
919D8101(810:1)	109	94	30
919D3000:1	112	97	33

The speed of each motor will be controlled using PWM. A Velleman K8055 Interface Board is used to enable the motor to be controlled by the motherboard via the USB port and an H-Bridge circuit.

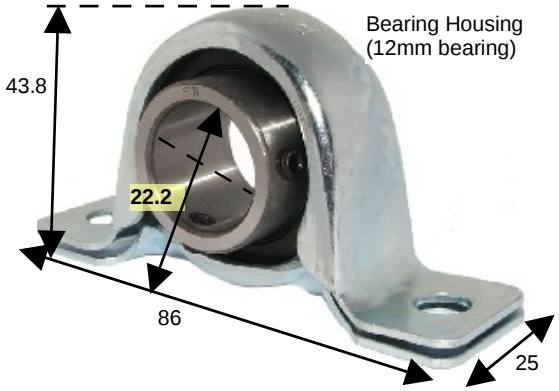
Chassis Mounting of the Motor

The motor must be mounted on the chassis in such a way that no large downward load is placed on the motors drive shaft. This can be achieved using a bearing.

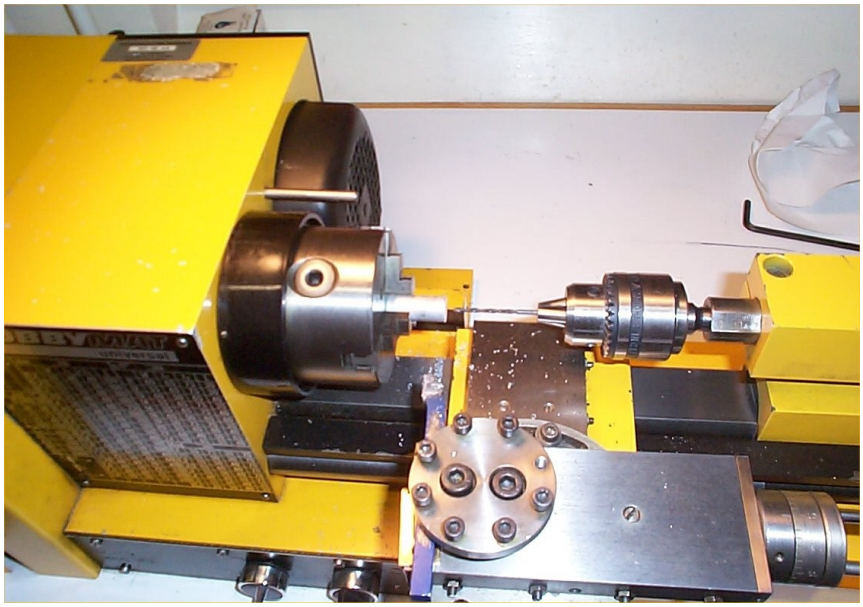




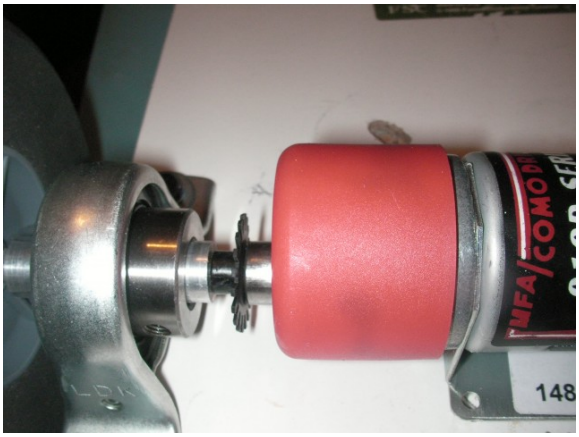
Pillow Housing for the bearing.

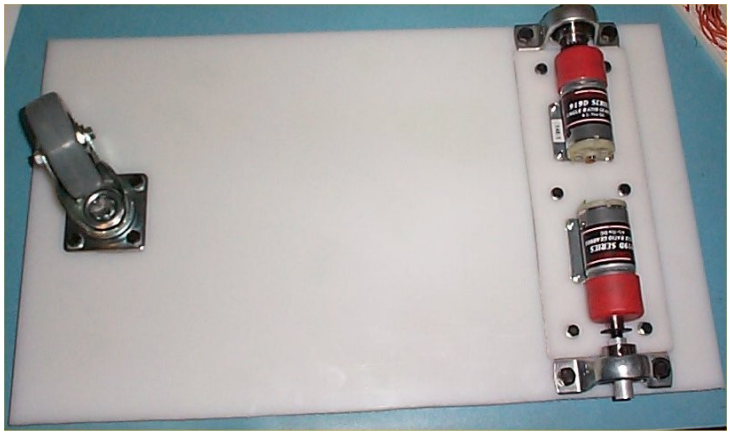


Making the Coupler on a mini-lathe



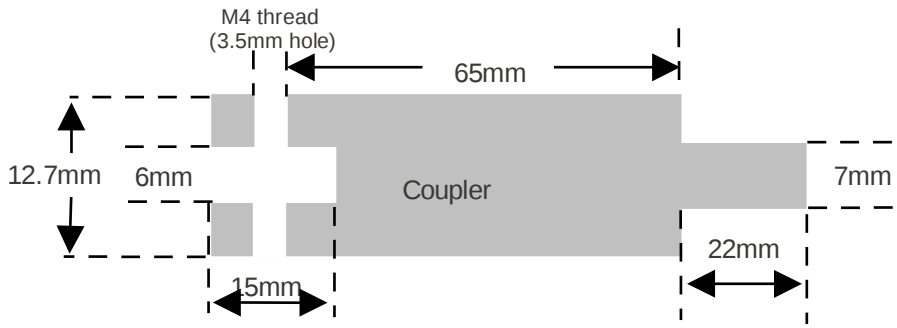
This is what it looks like assembled.





The wheel axle was pushed into the coupler and a grub screw used to hold it in place. I tested the two motors with my couplers by mounting them on to the chassis. I found that after a short time the grub screws came loose. Also the aluminium coupler is a bit soft to tighten up a screw in. I have changed the coupler so that the axle and the coupler are turned from the same piece of brass, removing the need for a grub screw on the wheel end of the coupler. I am also using a bigger grub screw to attach the coupler to the motor shaft.

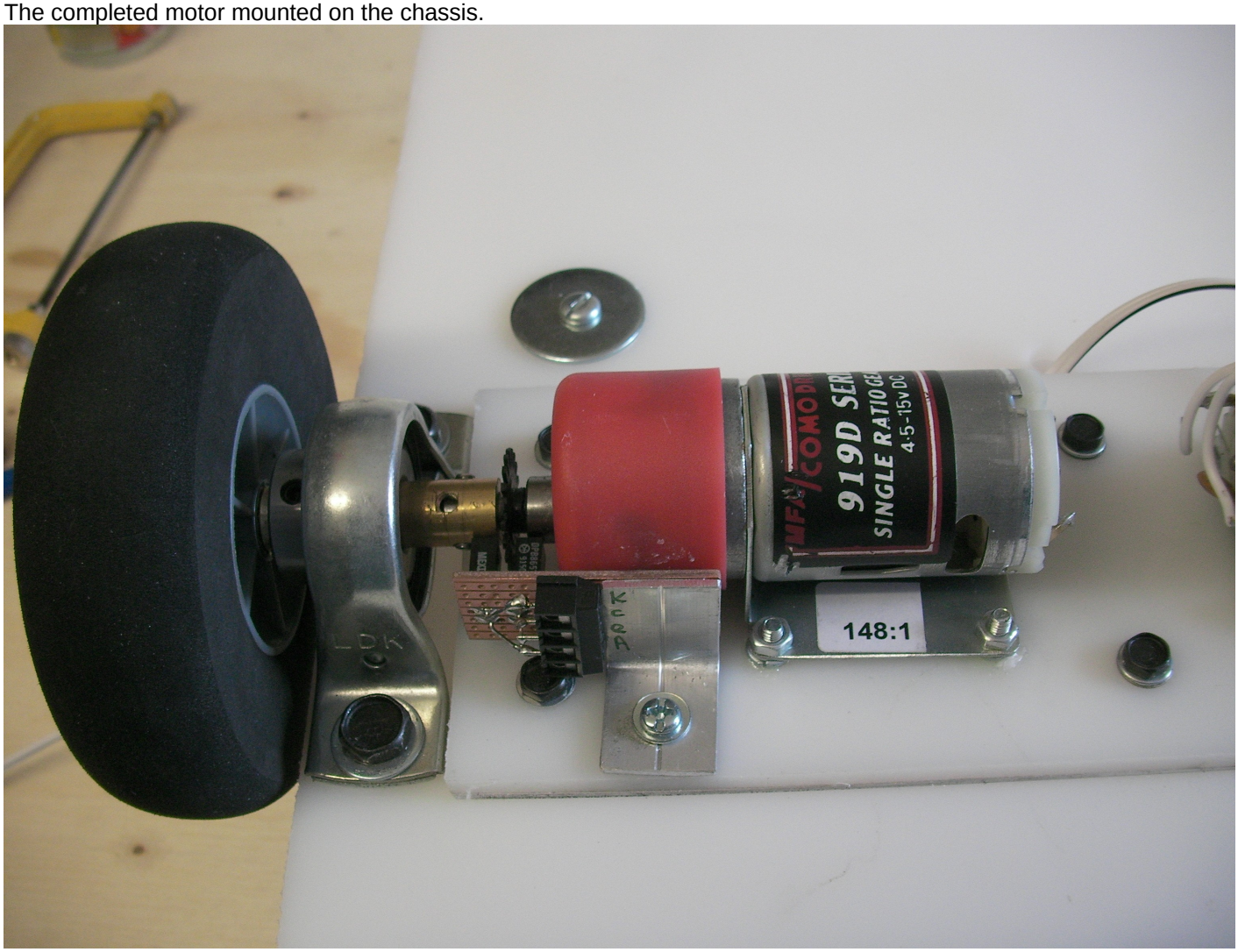
Improved Coupler Design



So far, this design works well

Make longer in diagram

The completed motor mounted on the chassis.



INTERFACING

Various means of interfacing with various input and output devices will be used.

USB Interface Board

This consists of a Velleman K8055 Interface Board with the following spec:

- 5 digital inputs (0= ground, 1= open)
- 2 analogue inputs with attenuation and amplification option
- 8 digital open collector output switches (max. 50V/100mA)
- 2 analogue outputs: (0 to 5V, output resistance 1K5)
- 2 PWM 0 to 100% open collector outputs max 100mA / 40V
- general conversion time: 20ms per command
- power supply through USB: approx. 70mA

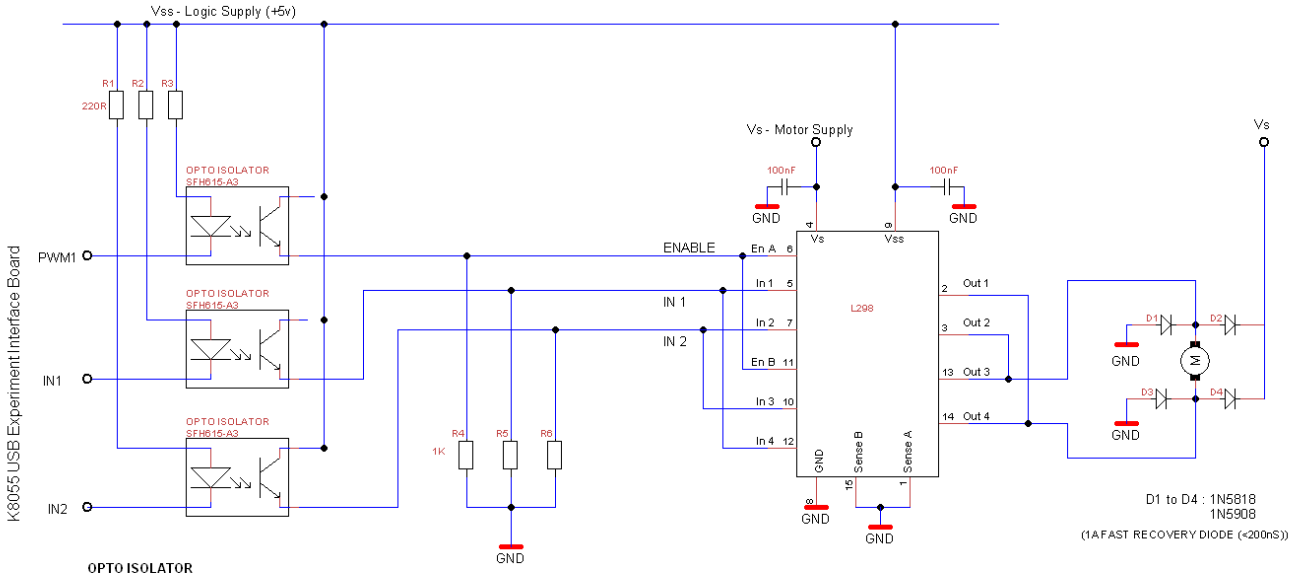
This board will be used for driving the two DC motors using PWM and receiving feedback from the two wheel encoders.

Interface Board Connections

Connector	I/O Type	Description	Notes
A1	Analogue Input	Battery Voltage	Battery Voltage (Motherboard & Logic)
A2	Analogue Input	Battery Voltage	Battery Voltage (Motor)
I1	Digital Input	Wheel Encoder 1	Counts the output of the wheel encoders to determine how far each wheel has moved
I2	Digital Input	Wheel Encoder 2	
I3	Digital Input		
I4	Digital Input		
I5	Digital Input		
PWM1	PWM Output	MOTOR 1 ENABLE	H-BRIDGE ENABLE for motor 1
PWM2	PWM Output	MOTOR 2 ENABLE	H-BRIDGE ENABLE for motor 2
CLAMP	External Voltage Supply for digital outputs (5 to 30V)	Not Used.	
O1	Digital Output	MOTOR 1 IN1	Motor 1 Direction
O2	Digital Output	MOTOR 1 IN2	Motor 1 Direction
O3	Digital Output	MOTOR 2 IN1	Motor 2 Direction
O4	Digital Output	MOTOR 2 IN2	Motor 2 Direction
O5	Digital Output		
O6	Digital Output		
O7	Digital Output		
O8	Digital Output		

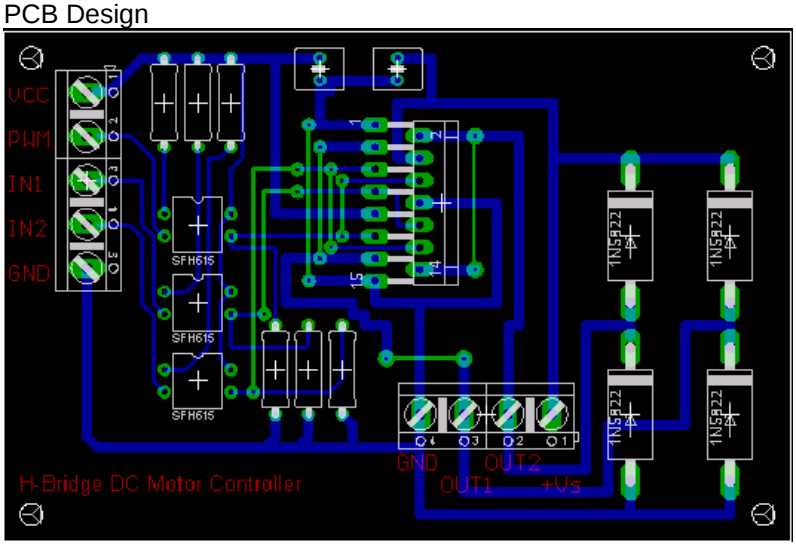
H-BRIDGE MOTOR DRIVER

The circuit below is sufficient to drive one motor up to 3 Amps. The circuit uses three outputs from the K8055 interface board : PWM1, O1 & O2. O1 and O2 control the direction of the motor. I.E forward, reverse or stop.
The PWM1 controls the motors speed. Also if PWM is set to Zero, the motor will free-wheel. The circuit uses the LM298N H-Bridge Driver IC.

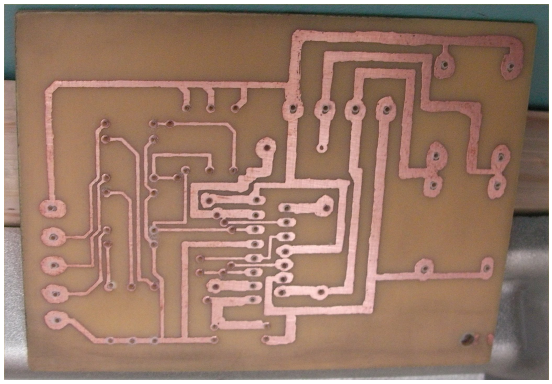


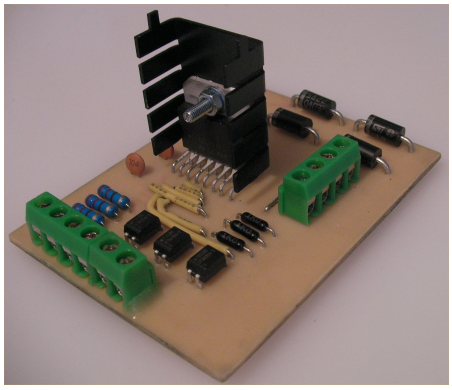
The above circuit is repeated for the second motor

PCB Design



Produced PCB





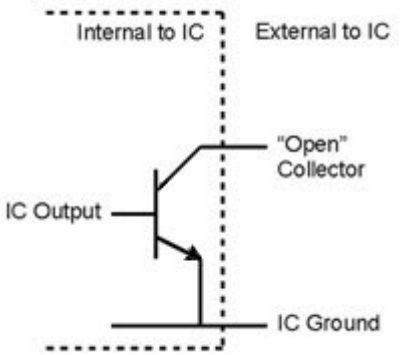
Possible Combinations of Inputs for ONE motor

Inputs			Result on Motor
PWM (ENABLE)	IN 1	IN2	
1	H	L	Forward
1	L	H	Reverse
1	H	H	Fast Motor Stop
1	L	L	Fast Motor Stop
0	X	X	Free Running Motor Stop

The PWM signal should be applied to the ENABLE input of the H-Bridge. IN1 and IN2 will be logic level inputs. To drive the motor, we set the direction using IN1 and IN2. ENABLE should then be switched between High and Low to produce a Pulse Width Modulated Signal. This will alternate the H-bridge between being driven and 'Free Running'. For maximum speed make ENABLE high all the time. In practice, this is achieved by using one of the PWM outputs of the K8055 interface board.

PWM Open Collector Output

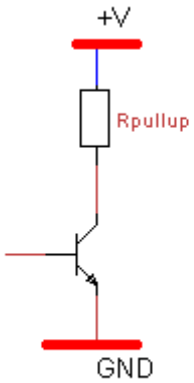
The outputs of the K8055 interface board are Open-Collector.



An open collector is just a Transistor with the emitter grounded. The collector is left unattached or "open" to anything hence it is called an open collector.

When there is no voltage applied to the base of the transistor, no current can flow from the collector to the emitter. When a voltage is applied to the base, the transistor turns on, allowing current to flow from the collector straight through to the emitter and to ground.

Note : If a voltage is connected to the collector, when the transistor is turned on, that voltage is going to flow from the collector to the emitter to ground. Thus creating a short circuit, probably destroying the transistor



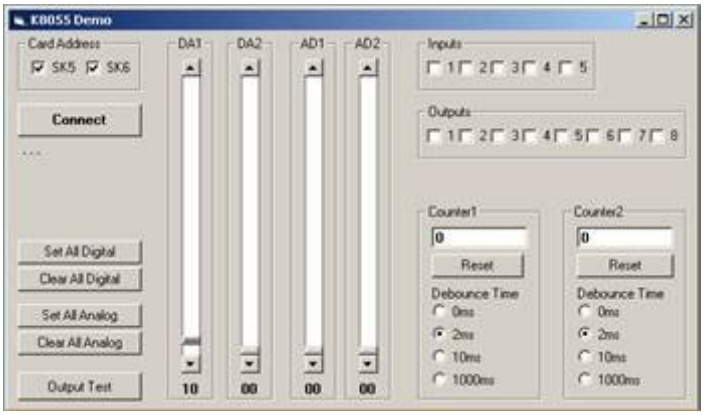
By using a Pull-up resistor we can limit the current flowing through the transistor.

Transistor Current (Ic) = V / R_{pullup}

The spec for the K8055 board states a maximum collector output of 100mA at 40V.

If we use a 220R pull-up resistor at 5V, the current will be 5/220 = 23mA. This is well within the spec of 40mA.

Test Software Screenshot

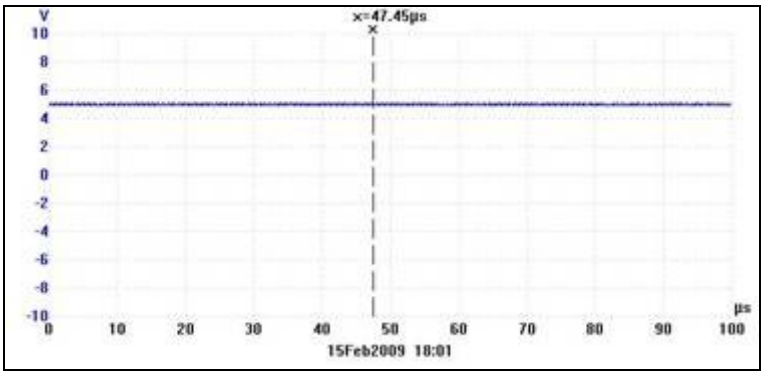


Some test software was provided with the K8055 card. It allows the testing of all the I/O's of the card.

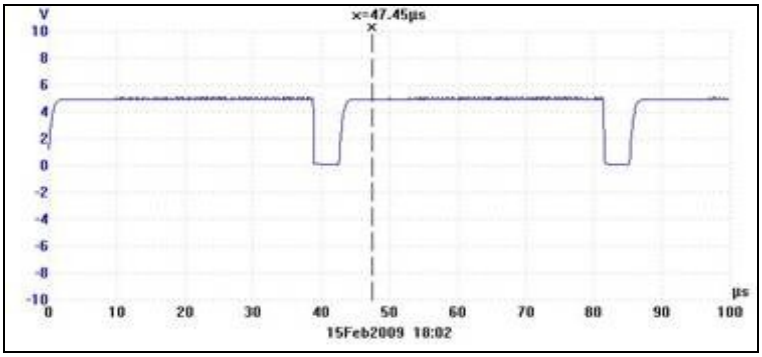
The two PWM outputs are linked to DA1 and DA2 respectively. As the digital to analogue converter is 8-bit, the PWM can be adjusted over 255 levels.

A value of 220R was used for R_{pullup}
The amount of PWM is adjusted by varying the output of the digital to analogue convert (DA1 for PWM1)

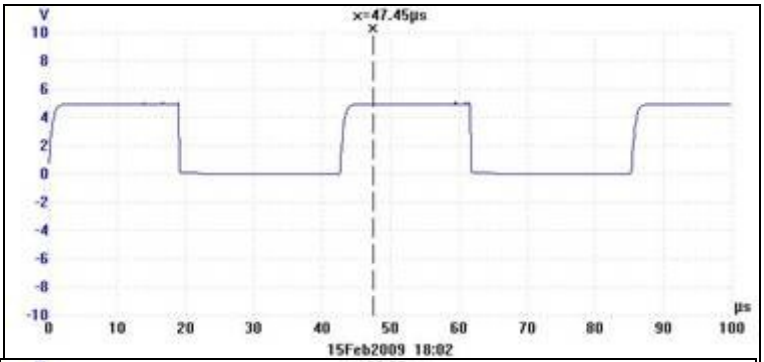
Measured PWM board outputs



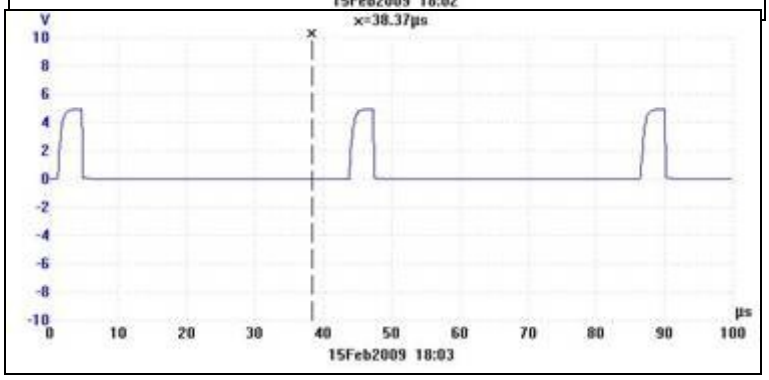
DA1 set to 0. PWM is 100%.
Full power will be provided to the motor.



DA1 set to 10. PWM is approx 90%.
90% power will be provided to the motor.



DA1 set to 128
PWM is approx 40%



DA1 set to 220

PWM is approx 5%
Only 5% power will be provided to the motor

So for max PWM DA1 should be set to 0. As DA1 increases the PWM decreases.

MOTOR ENCODERS

An encoder is used to measure the rotation of each wheel. The speed of each motor is controlled by the computer using a PWM signal to each motor. Without using an encoder, it will be impossible to know how fast each wheel is turning. A transmitter and receiver sit either side of an encoder wheel. As the drive shaft rotates, the encoder wheel will also rotate. This will cause the beam from the transmitter to the receiver to be repeatedly interrupted at a rate dependant on the speed of rotation.

My encoder design is very simple. The main parts are:

Infra-red Transmitter and Receiver

These were obtained from ebay

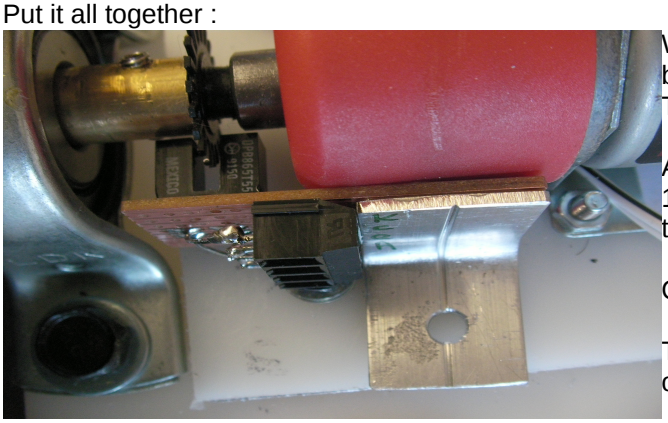
Transmitter Spec (IR LED):	Maximum forward current = 100mA Luminous Intensity = 1.8mW at 50mA Beam angle = 30°
Receiver Spec (Photo transistor):	VCEO 30V Maximum Dissipation: = 100mW Ic Max = 20mA Viewing angle = 70°

Encoder Wheel

This is mounted on the drive shaft of the motor.
The wheel has 20 fins.

The encoder wheel was salvaged from a couple of old Syringe Drivers that were being scrapped by my hospital, which is also my employer.

Put it all together :



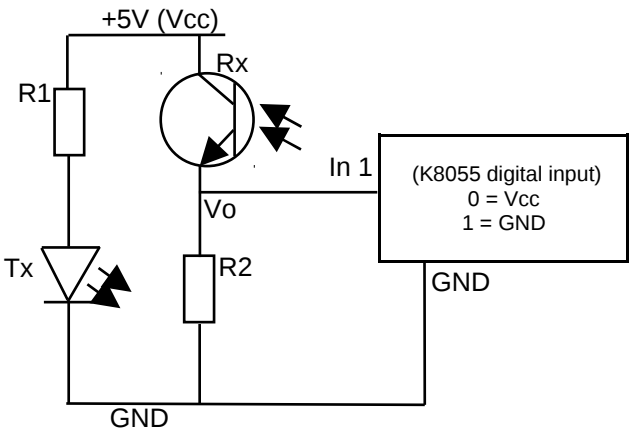
When the drive shaft rotates, a series of electrical pulses will be generated by the IR Receiver as the beam from the IR Transmitter is interrupted by the encoder wheel.

As the encoder wheel has 20 fins, each pulse will represent 18° rotation of the wheel. After 20 pulses, we will know that the wheel has rotated once.

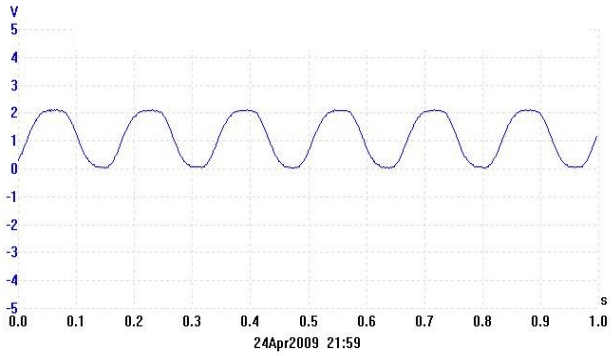
Obviously if the wheel skids, this will give rise to false pulses.

The output of the receiver is conditioned and then fed into one of the digital inputs of the K8055 Interface Board.

Circuit Design



Measured Vo with encoder wheel turning



Transmitter

R1 is to limit the current flowing through the Transmitter (Tx). The maximum forward current If=50mA.
Max Power dissipation = 100mW

R1=330 Ohms
V=5V
If = 5/330 = 15mA. Power = V.I = 5 * 15 = 75mW This is well within the transmitters specification.

Receiver

The receiver (Rx) is a Photo Transistor, therefore the base generates a voltage when exposed to IR light. This causes the resistance of the receivers collector emitter junction (Rr) to vary depending on the level of IR received.
R2 is the bottom half of a voltage divider. A 4.7KOhm resistor was chosen for R2:

Output Voltage Vo = (R2 / (Rr +R2)) . Vcc
Rx Collector Current Ic = Vcc / (Rr+R2)

When the receiver is not illuminated by the IR Transmitter Vo will be zero. When illuminated by the IR, Vo will rise to approximately 2V. Verification of this is shown in the scope capture above.

Input to the K8055 Interface Board

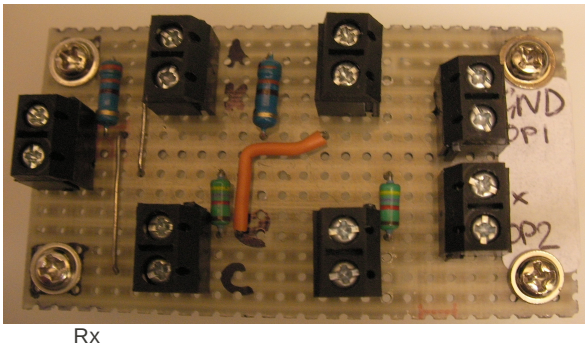
A positive voltage on the digital input to the K8055 will be recognised as an Open switch
A near GND voltage (<0.7V approx) on the digital input will be recognised as a Closed switch.

Strip-board design

The circuit was constructed using stripboard. The board shown below has two circuits on it, one for each encoder.

Tx
Anode & Cathode
connector

5V
Supply



Rx
Collector & Emitter
connector

Testing THE MOTORS / ENCODERS

Initial testing of motors and encoders.

At this point the following had been completed:

1) Inner Chassis construction
2) Motors and wheels mounted
3) Encoders built and assembled on each motor
4) ITX motherboard mounted on chassis and WIN98 installed
5) Visual Basic 6 installed
6) Velleman K8055 USB Interface Board installed
7) H-Bridge motor controlled and 5v PSU constructed and mounted on robot

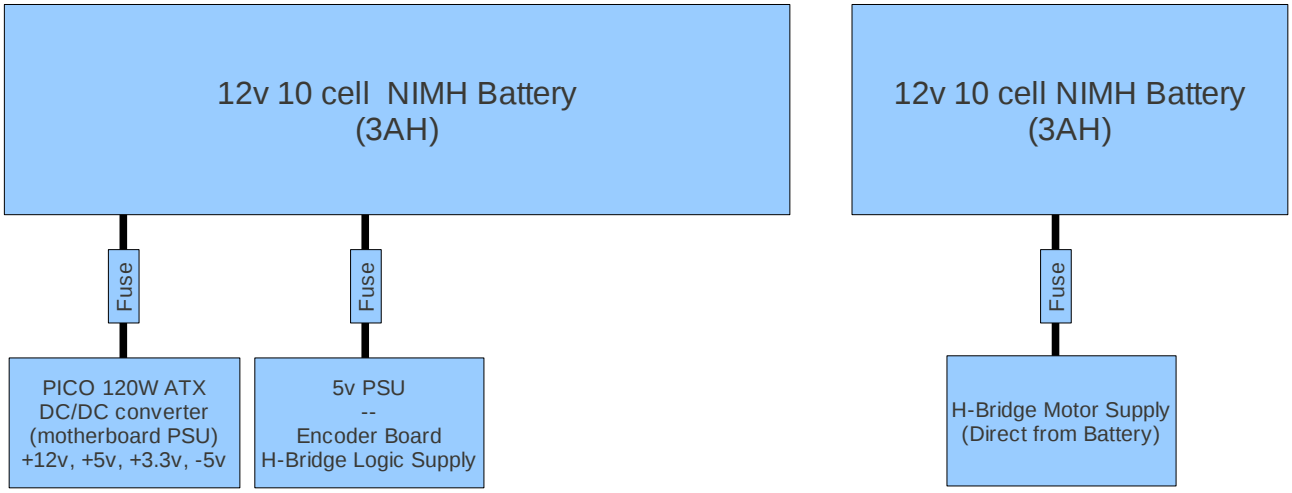
The robot was now ready for some initial testing.

Testing consisted of making sure that the computer could control the robot via the USB port, as well as receive information from the encoder on each wheel. A simple VB6 program was written that controlled the speed of each wheel, and took real time inputs from each wheel encoder. The program then made slight adjustments to the speed of one of the wheels to ensure that the robot moved in a straight line. This approach seems to work well. I also experimented in getting the robot making set movements. IE, Forward for 5 seconds, reverse, turn right, turn left.

I am using differential drive to turn the robot. IE to turn left the left wheel reverses and the right wheel goes forward .

POWER SUPPLIES

Power requirements for the robot are currently as follows:



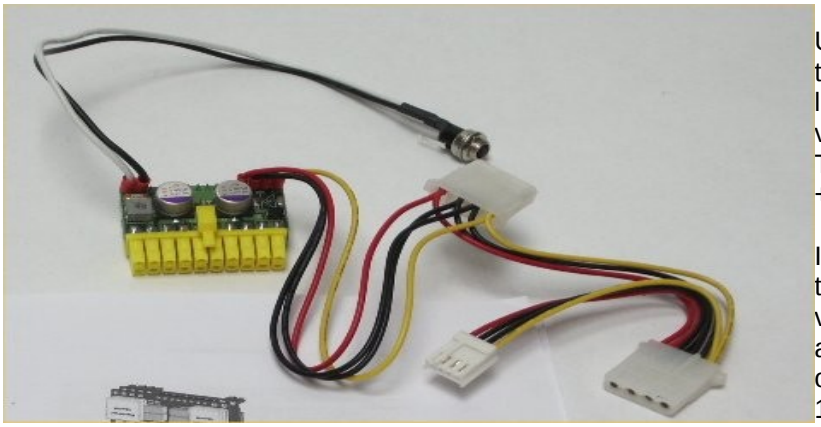
BATTERIES

Two batteries are currently being used:
Motherboard and logic circuits : 10 cell 12V 3Ah NIMH Battery.
Motors : 10 cell 12V 3Ah Lead Acid Battery.

One of the advantages of the NIMH battery over using a lead acid battery is a higher capacity vs size and hence lower weight.

Power Supply for VIA EPIA Mini-ITX Mainboard

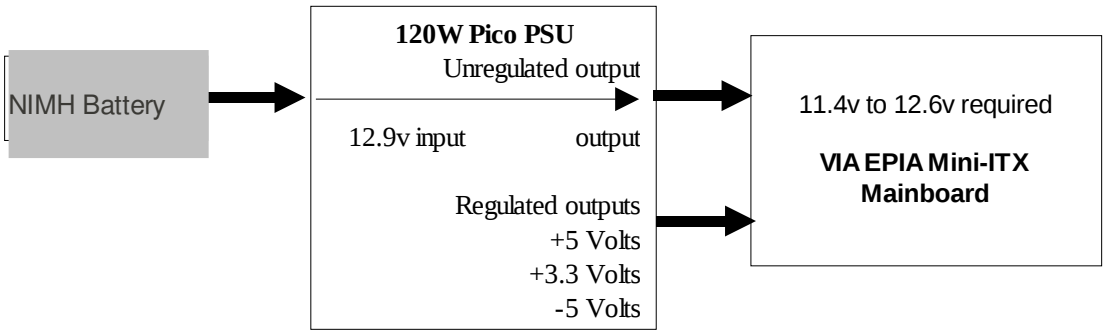
The motherboard is powered using a Pico 120W ATX DC-DC converter, which requires a nominal 12V input and provides the necessary outputs for the motherboard and accessories.



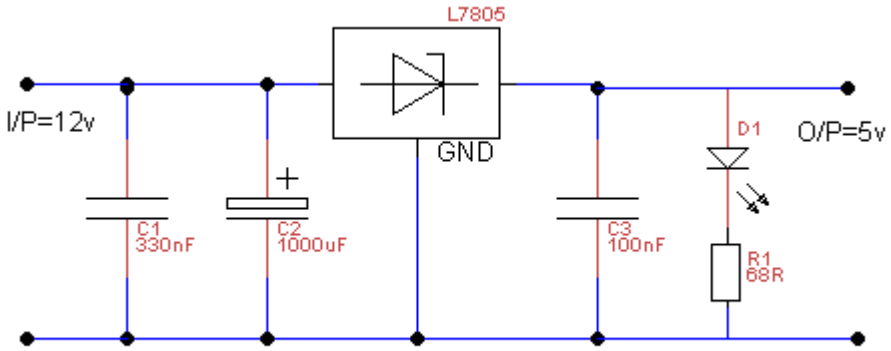
Unfortunately the 12V line is just switched through the DC-DC converter, so for the 12V line the voltage output will be equal to the voltage input minus approx 0.2 Volts.
The ATX specification for the 12V line is 12V +/- 5%. Or 11.4V to 12.6V

I'm using a 10 cell 12V NIMH battery to power the motherboard. When fully charged the voltage supplied to the motherboard is approximately 13.2V (under load). This is just outside the maximum specification value for the 12V line,

For more information see Appendix 2.



5v PSU



C1: 330nF Ceramic
C2: 2200uF 25v Electrolytic
C3: 100nF Ceramic
D1: Super Bright Pink LED > Vf=3.4v, If=25mA
R1: 68R resistor

This board supplies the following circuits:

- Wheel encoder board
- H Bridge logic supply
- Servo Controller
- Pan & Tilt Servo Motors Power

The design uses an LM7805 linear regulator. This regulator has current limiting and thermal shut-down making it hard to damage. With a heat-sink it can give over 1 A output current.

C1 and C3 filter out any high frequencies, which may have an effect on any logic or sensitive circuits. These are recommended on the data sheet.
C2 will reduce any sudden drops in input voltage, which could effect the regulator, although as the input to the regulator is a battery this capacitor may not be required.

LED Resistor calculation:

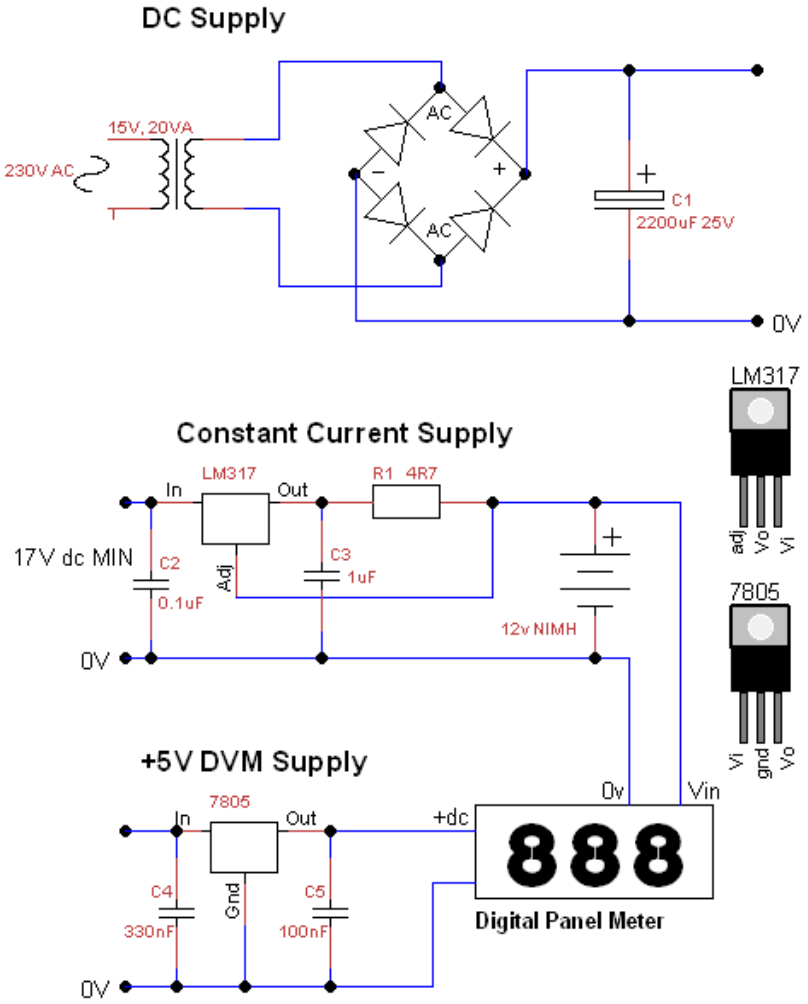
Supply Voltage (Vs) =5v
R= (Vs – Vf) / If = (5 – 3.4) / 0.025 = 64 ohms.
Closest resistor = 68 ohms

Battery Charger

This is required to charge the two batteries used on the robot.

The simplest way of charging a NIMH battery is to charge at C/10 or less (10% of the batteries rated capacity per hour). So a 3000mAH battery would be charged at 300mA for 15 hours. At this rate, the battery can't be over charged.

This circuit is suitable for charging a 9 or 10 cell 3AH NIMH battery pack. As the charge rate is less than 0.1C (300mA) the battery should be charged over a period of approximately 15 hours. The Digital panel meter enables the battery voltage to be monitored while charging. A 10 cell battery pack will be fully charged when the voltage reaches approximately 14.1V (1.41V per cell)



DC Supply
A 15V transformer is rectified and smoothed using a 2200uF capacitor. Alternatively a 17v regulated power supply can be used in place of the DC Supply.

Constant Current Supply
This circuit is based on the LM317 regulator. The input voltage needs to be at least 3V higher than the battery voltage. C2 & C4 limits high frequency noise on the input to each regulator. C3 and C5 stop any oscillations on the output of the regulator.

Battery Current Setting
The regulator will always set the voltage between the Adj pin and the Out pin to be 1.25 Volts (Across R1). If the current changes through the battery and hence the resistor (R1) , then the voltage between Adj and Out will also change. The regulator will change the output voltage to maintain the 1.25V, and thus the current will be held constant.

The voltage between Adj and Out is always 1.25v (Vref)

Therefore the resistance (R1) to give current (Iconst) = Vref/Iconst = 1.25V/0.3A = 4.2 Ohms.

At 300mA charge current, the battery pack gets quite warm when fully charged, so I decided to lower the charging current a little.

Using a 4.7 Ohm resistor: Iconst = Vref / R = 1.25/4.7 = 0.266A

5V DVM Supply
This provides the power for the digital panel voltmeter and is based on a 7805 +5Volt regulator.

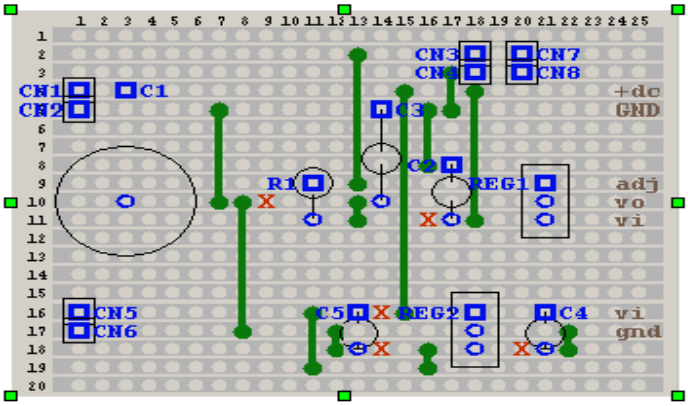
Testing
The total current drawn by the circuit, measured by placing an ammeter in series with the output of the DC supply was 330mA.

I measured the output current of the circuit for three conditions.
Short across output with battery removed
Flat battery in circuit and charging (11.5V)
Almost fully charged battery (14.0V)

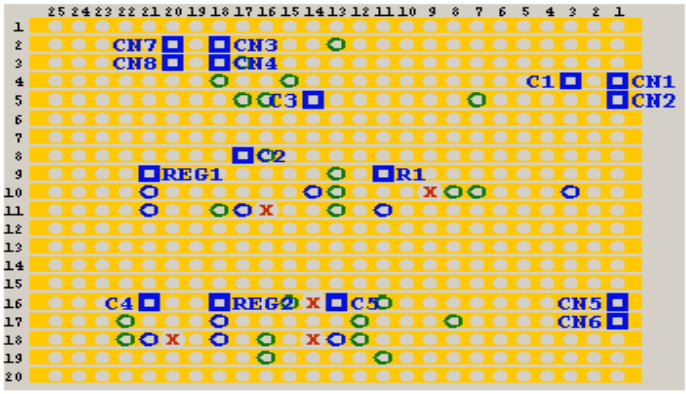
In each case I measured a current of approximately 264mA. This compares favourably with the calculated value of 266mA. The measured voltage between Vout and Adj of the LM117 was 1.25V as expected.

Stripboard Layout

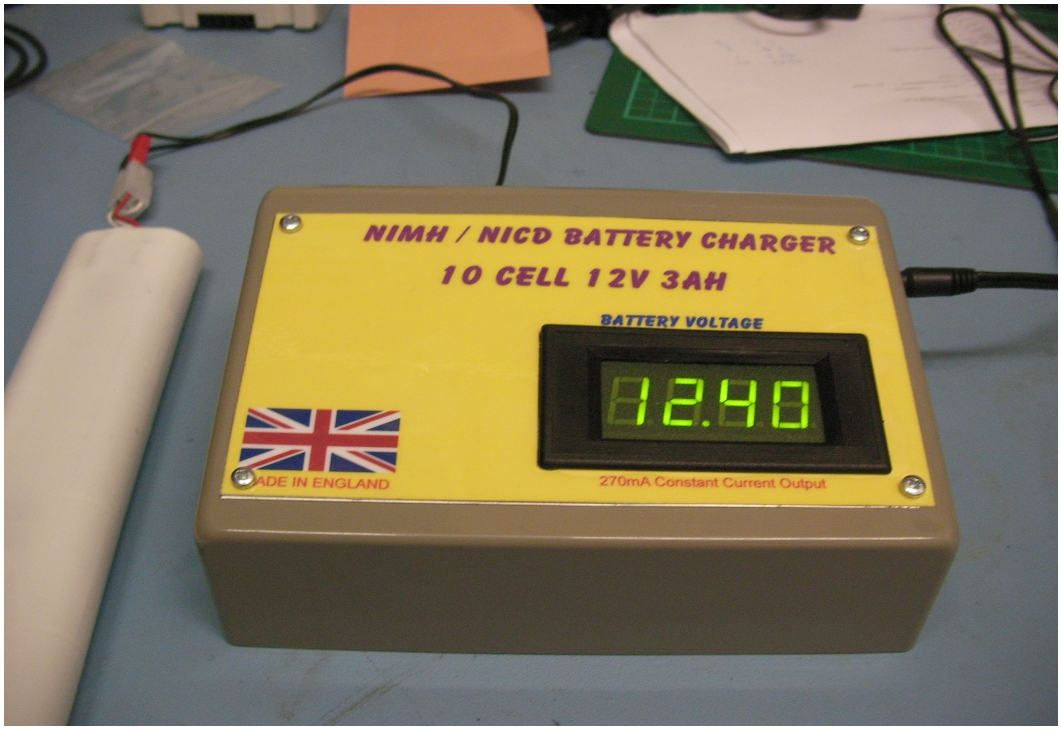
Top - Board View



Bottom – Track View



The finished Charger



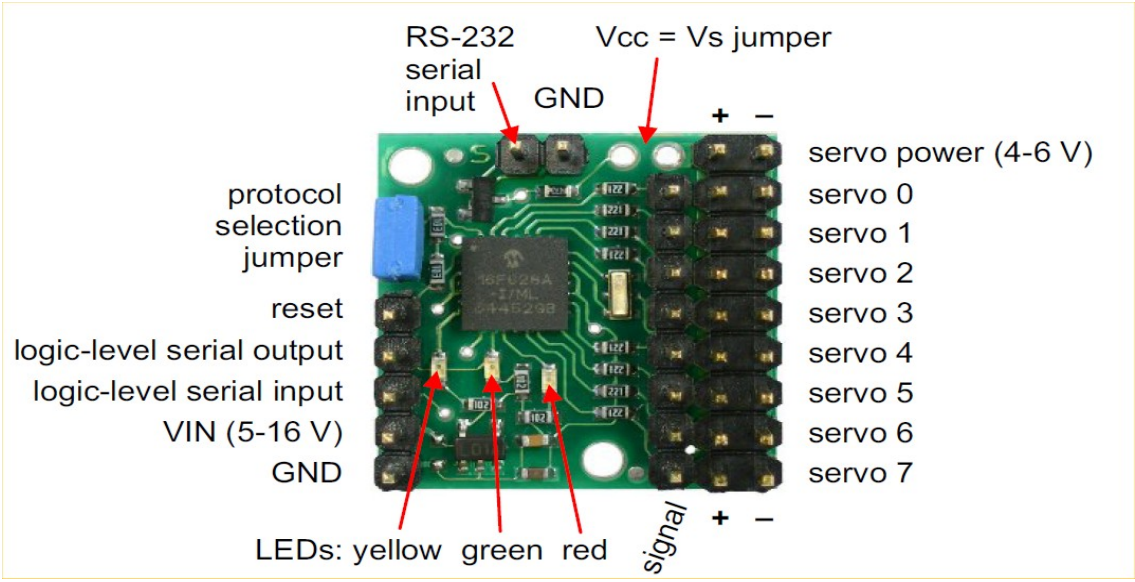
PAN AND TILT HEAD

For controlling the robot's head I chose a Pan and Tilt head from Lynxmotion. (www.lynxmotion.com)
The head uses two Hitec servo motors.

Specifications

Voltage = 4.8 - 6.0vdc
Torque = 57 oz.-in.
Weight = 1.66 oz.
Speed = 0.16s / 60 degrees
Standard Size = 1.6" x 0.8" x 1.4"
Range = 180°

To control the head, I decided to use a Pololu (www.pololu.com) Micro Serial Servo Controller. This device connects to the serial port of the motherboard and can control up to 8 servos.

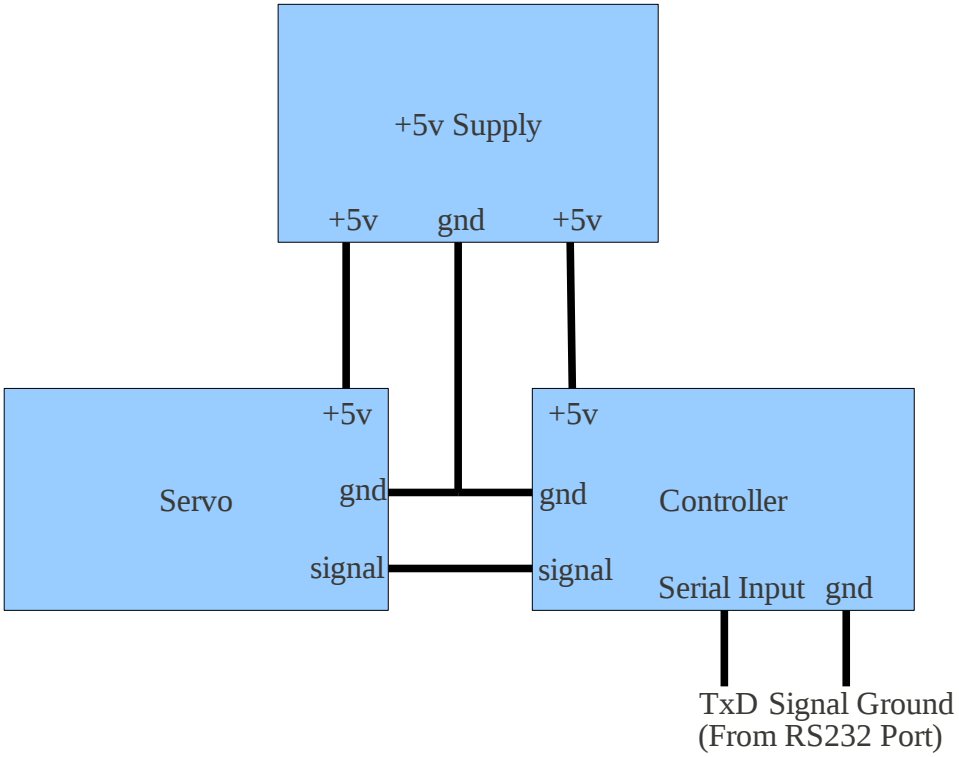


Connections to the serial port of the motherboard are very simple. The controller operates in Simplex mode, which means it just needs the TxD and Signal ground pins of the serial port connected to the controller.

Serial (RS232) port interface pin-out and signals

Pin No	Acronym	Full name	Description
3	***TxD***	***Transmit Data***	***Transmits bytes from the PC***
5	SG	Signal Ground	

Connection Diagram



The controller board and the two servo motors are all powered from the 5V supply mounted on the main robot chassis.

The servo is programmed using Visual Basic 6. To program the servo controller a simple protocol is used. The example VB listing below shows the basics of this.

On creating the VB project, I added an MSComm component to the VB form and set the baud rate to 9600.

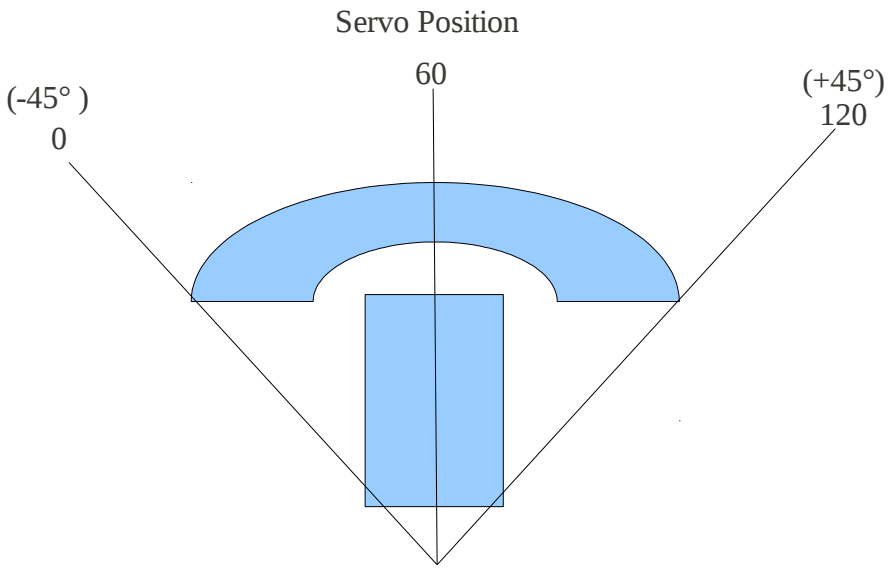
```
Dim Buffer(&H4) As Byte          // Create a buffer to hold the data to be sent to the serial port
Buffer(0) = 128                  // Synchronisation value always 128
Buffer(1) = 1                    // Device type number always 1 for the Micro Serial Servo Controller
Buffer(2) = 1                    // Command > 1=speed, 2=position
Buffer(3) = 1                    // servo number
Buffer(4) = 8:                   // 7 bit data (127 Max)
MSComm1.Output = Buffer          // send the data to the serial port
```

In the above example the speed of servo number 1 will be set to 8

```
Buffer(0) = 128                  // Synchronisation value always 128
Buffer(1) = 1                    // Device type number always 1 for the Micro Serial Servo Controller
Buffer(2) = 2                    // Command > 1=speed, 2=position
Buffer(3) = 1:                   // servo number
Buffer(4) = 60:                  // 7 bit data (127 Max)
MSComm1.Output = Buffer
```

In the above example the position of servo number 1 will be set to 64. This is the mid position.

Using the above combination of Pan & Tilt head, Micro serial servo controller and VB6,it is very easy to control the movement of the robot's head.



The servo scans over a range of ±45°.
The servo position is controlled by sending a numeric value between 0 and 120 to the servo controller. 60 is the mid position. . Servo position 0 corresponds to -45° and position 120 corresponds to +45°. So if the servo scans from 0 to 120 in steps of 5, this will correspond to 25 (3.75°) steps.

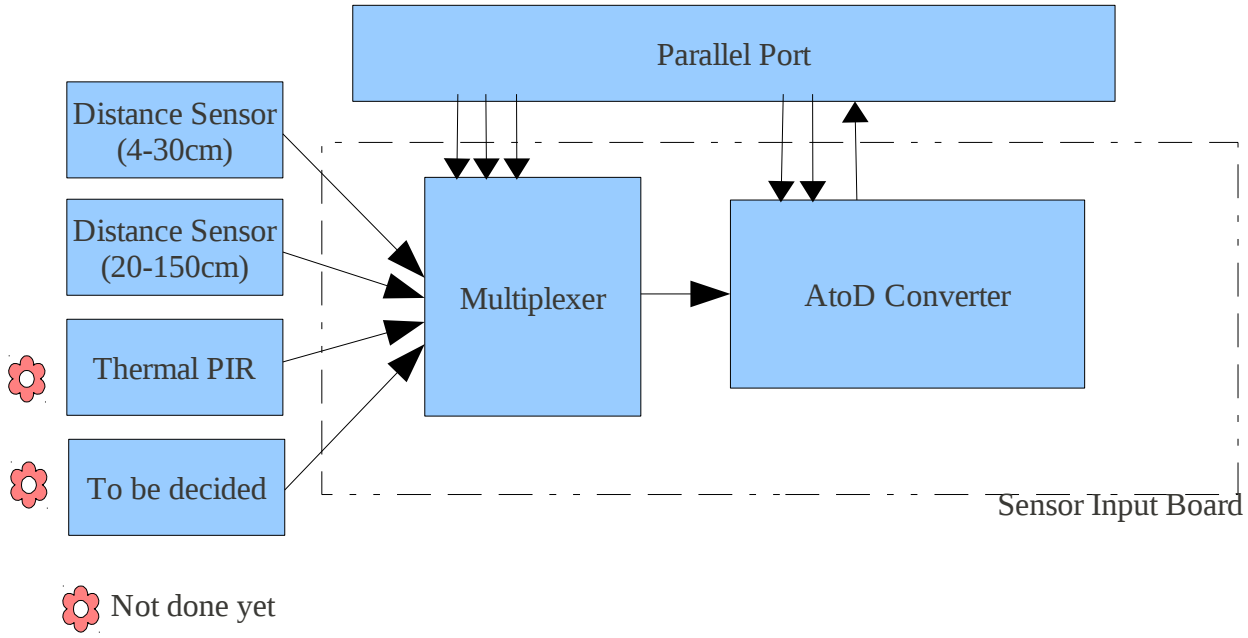
SENSOR INPUTS

Four Sensors will be used.

1) Sharp IR Distance Sensor (4-30cm)
2) Sharp IR Distance Sensor (20-150cm)
3) Thermal PIR sensor (Type to be decided)
4) To be decided

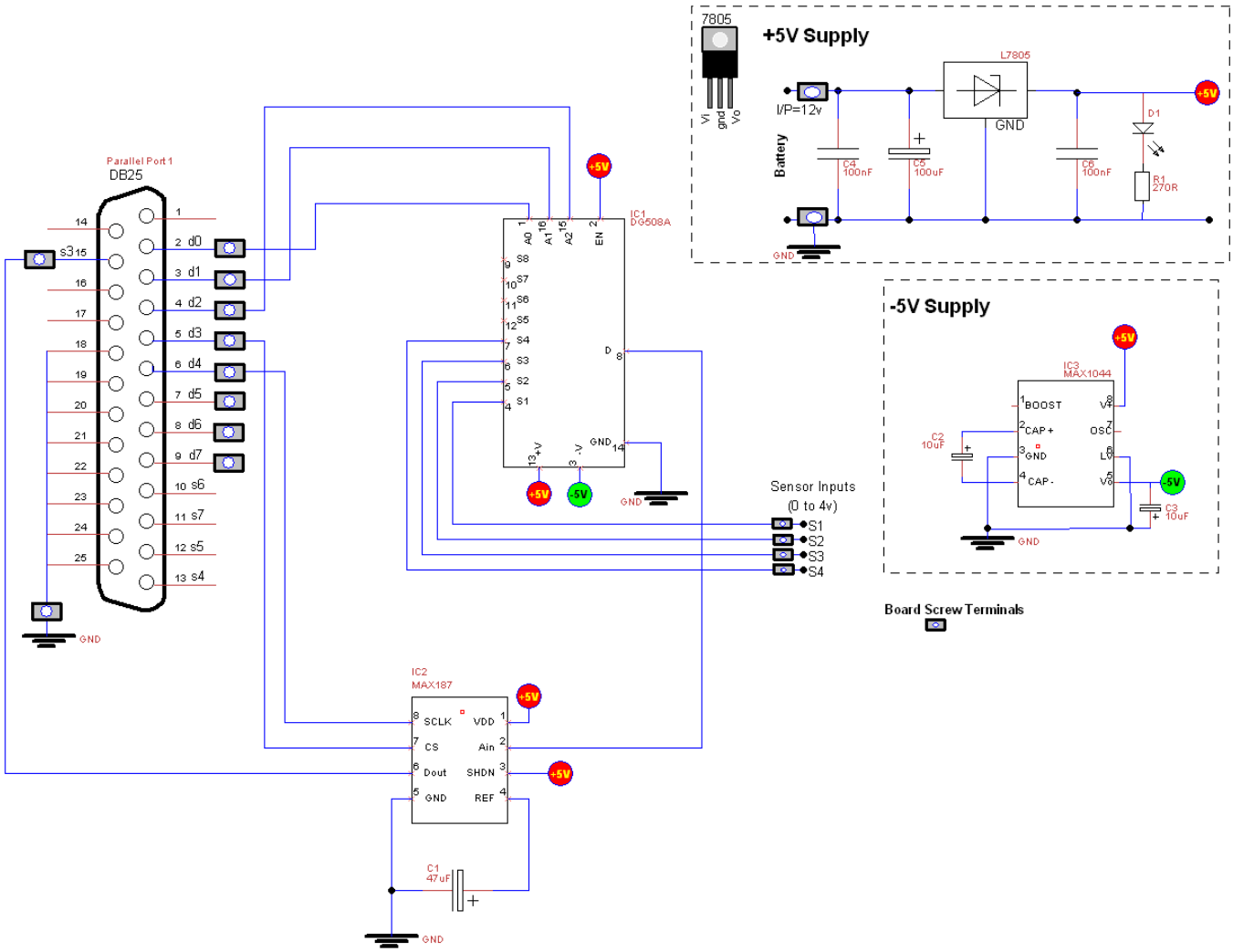
All sensors will be interfaced with the motherboard via the parallel port, using an AtoD converter. The output of each sensor will be multiplexed into a single input of the AtoD converter.

The basic block diagram of the circuit is shown below:



Sensor Input Board

This board will enable up to 4 analogue inputs to be measured via the parallel port up to a maximum of 4.096V each.



Sensors

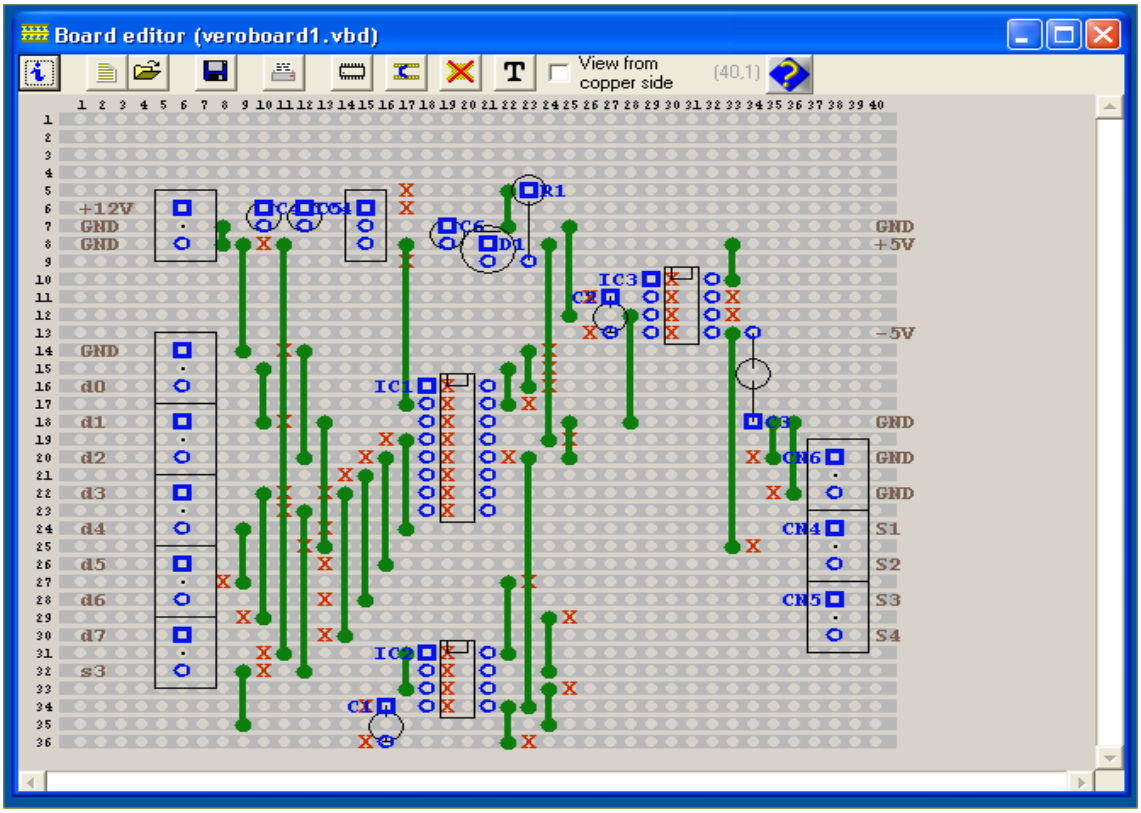
So far, the following will be used as sensors:
Infra-Red R Range finder 4-30cm (short range)
Infra-Red R Range finder 20-150cm (Long Range)
PIR thermal detector.

Brief Circuit Description and Function

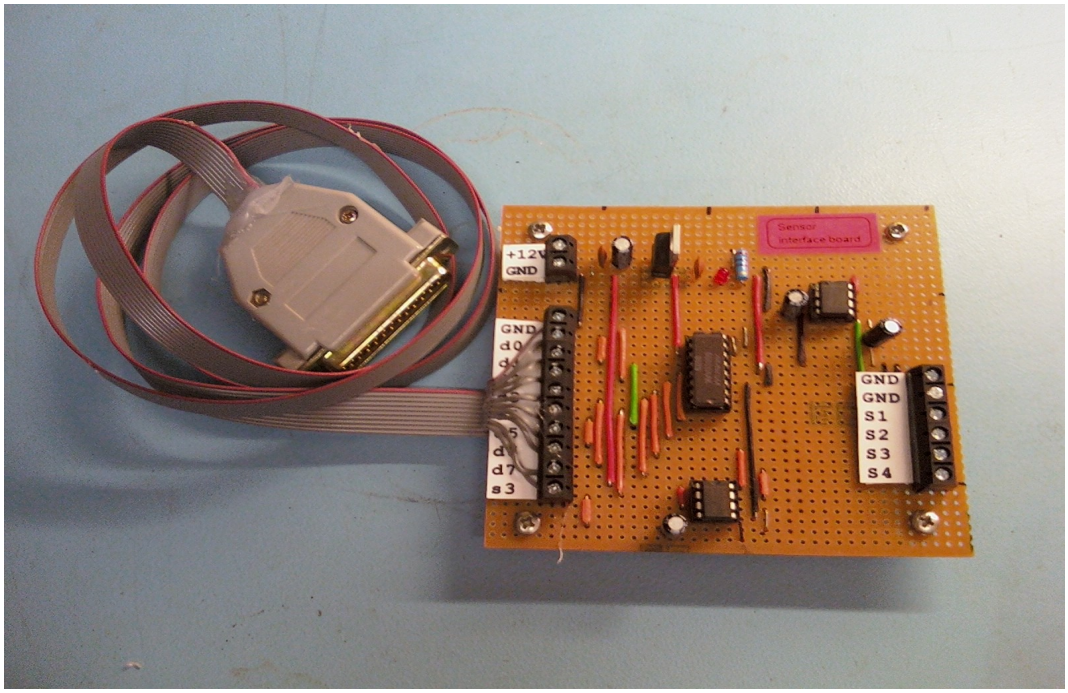
The circuit consists mainly of a MAX187 12 bit AtoD converter and a DG508A Multiplexer. The multiplexer needs a +/- V supply, so I am using a MAX1044 charge pump to generate -5V. +5V is supplied using a 7805 linear regulator. I've used this design for many other applications and it works well.

Sensor Input Board Strip-board Design

The strip-board layout was done using a program called Verodes
The 12V supply is derived from the Motherboard/PSU battery.



Completed Board ready for connection to the Parallel Port.



The whole circuit only draws 12.5mA from the 12V supply.

Parallel Port Connections

The Data Register and the Status Register of the parallel port are used to interface with the board. This can be implemented fairly easily using Visual Basic 6. The connections to the parallel port are shown below. These are described in more details in the following paragraphs.

Data Register - Parallel Port Output/Input Lines

Parallel Port Register	d0 (pin 2)	d1 (pin 3)	d2 (pin 4)	d3 (pin 5)	d4 (pin 6)	s3 (pin 15)	d% = data registers s% = status register
Circuit Board Connection	A0	A1	A2	$\overline{CS}$	SCLK	Dout	
Description	DG508A Multiplexor Input Select			MAX 187 AtoD			

Maxim DG508A Multiplexer

This will take up to 8 inputs via S1 to S8 and connect only one of them to its D output, according to the logic levels set on pins A0 to A2. I will only be using 4 of the inputs.

DG508A Truth Table

A2 A1 A0 EN ON SWITCH

X X X 0 NONE
0 0 0 1 S1
0 0 1 1 S2
0 1 0 1 S3
0 1 1 1 S4

More description of how the DG508 works. …..

DG508A Multiplexer Pin out

1,15,16	A0, A1, A2	Address Inputs (Input select)
2	EN	Enable. Take high to enable the IC
3, 13	V+, V-	Supply Voltage +/- 18V max
4,5,6,7,9,10,11,12	S1 to S8	Signal Input Lines
8	D	Switched Signal Output
14	GND	Ground (0v)

MAX 187 12bit Serial Analogue to Digital Converter

This converts the analogue 'D' output of the DG508A into a 12 bit digital signal, which is then input to the motherboard via the parallel port. Conversion-start and data-read operations are controlled by the $\overline{CS}$ and SCLK digital inputs. A $\overline{CS}$ falling edge initiates a conversion sequence and DOUT changes from high impedance to logic low. End of conversion (EOC) is signalled by DOUT going high. DOUT's. SCLK shifts the data out of the device any time after the conversion is complete. DOUT transitions on SCLK's falling edge. The next falling clock edge produces the MSB of the conversion at DOUT, followed by the remaining bits. Since there are 12 data bits and one leading high bit, at least 13 falling clock edges are needed to shift out these bits.

MAX 187 12bit Serial AtoD Converter Pinout

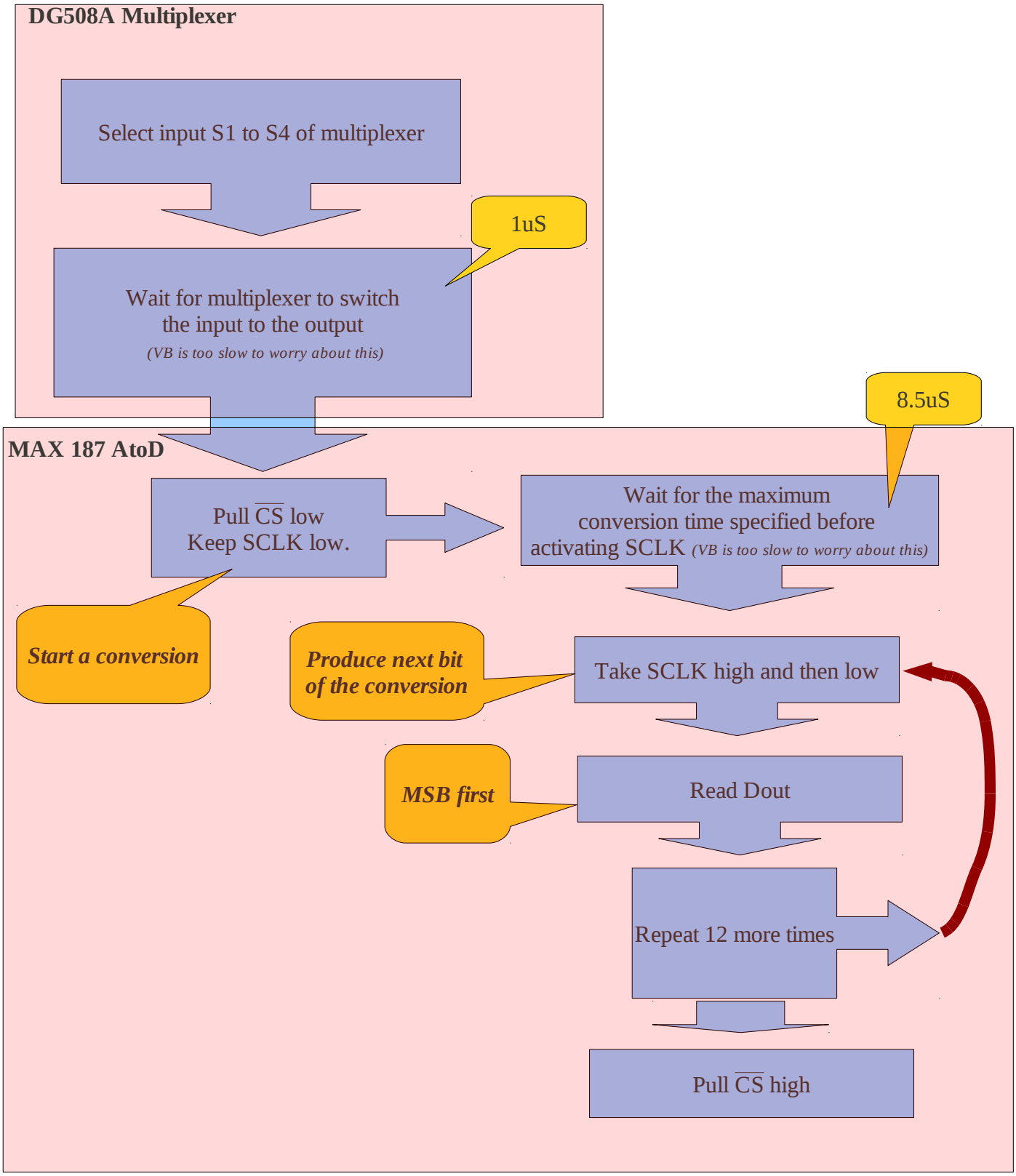
1	VDD	Supply voltage, +5V ±5%
2	Ain	**ANALOGUE IN** - Sampling analogue input, 0V to VREF range (4.096V)
3	SHDN	**SHUTDOWN** - Pulling SHDN high enables the internal reference
4	REF	Reference voltage - Sets analogue voltage range to 4.096V.
5	GND	Digital ground
6	Dout	**DATA OUT** - Serial data output. Data changes state at SCLK's falling edge.
7	$\overline{CS}$	**CONVERSION START** - Active-low Chip Select initiates conversions on the falling edge. When CS is high, DOUT is high impedance.
8	SCLK	**SERIAL CLOCK INPUT**. Clocks data out with rates up to 5MHz.

Only four connections are required to the parallel port on the motherboard. These are:

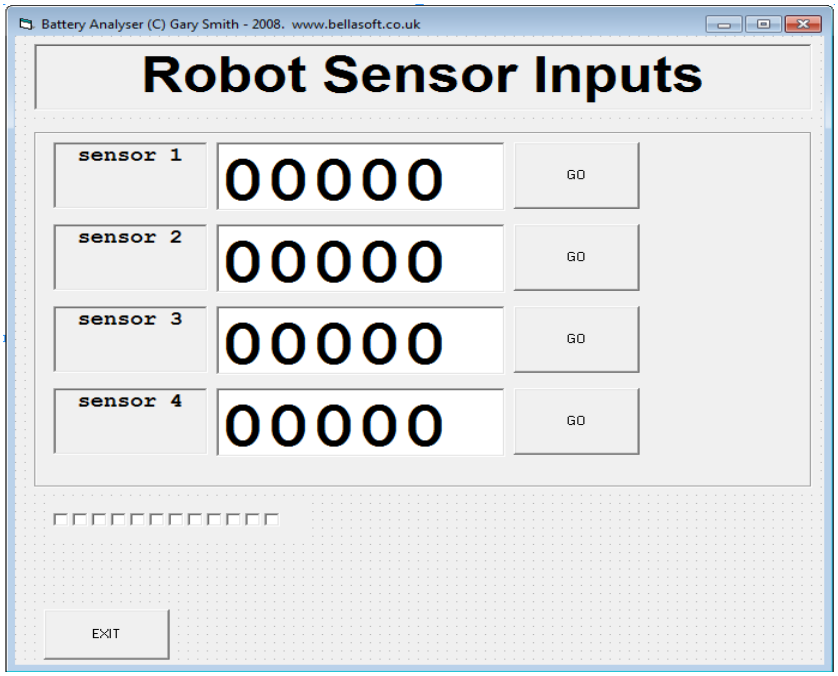
MAX187 Pin	Parallel Port Pin
SCLK	d4 (Pin 6)
$\overline{CS}$	d3 (Pin 5)
Dout	s3 (Pin 15)
GND	GND (Pins 18 to 25)

The ADC is a 12 bit Serial ADC with a built in reference voltage of 4.096V. 12 bits equates to 4095 digital levels. Therefore the resolution of the device is 4.096/4095 = 1.0mV / bit.

Simple Programming algorithm for Multiplexer and AtoD Converter



Sensor Input Test Program



This program (See Appendix 4) selects one of the inputs to the multiplexer (S1 to S4) by outputting a 3 bit value d0, d1 & d2 via the parallel port to the address select pins of the multiplexer (A0, A1 and A2).

The selected input is passed out of the multiplexer(D) to the input of the AtoD converter(Ain). This is then converted to a 12 bit digital signal which can then be read by the parallel port via the status port (bit s7)

To test the sensor inputs, I applied a known voltage (1.2v) to (S1 to S4). By clicking each GO button I got the software to measure the applied voltage. As stated earlier, the AtoD is 12 bits with a reference voltage of 4.096v. 12 bits equates to 4095 digital levels. Therefore the resolution of the device is 4.096/4095 = 1.0mV / bit. So if I apply say 1.20 volts I should get a value of 1200 from the AtoD. So 1mV = 1 bit output from the AtoD.

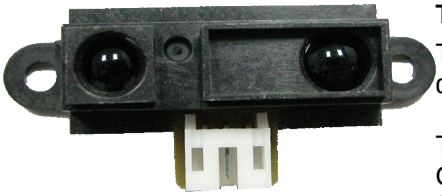
Test results

Applied Voltage (Volts)	Measured Voltage (mV)

DIAGRAM OF SENSOR BOARD MOUNTED ON ROBOT CHASSIS

Sharp IR Distance Sensor (20-150cm)

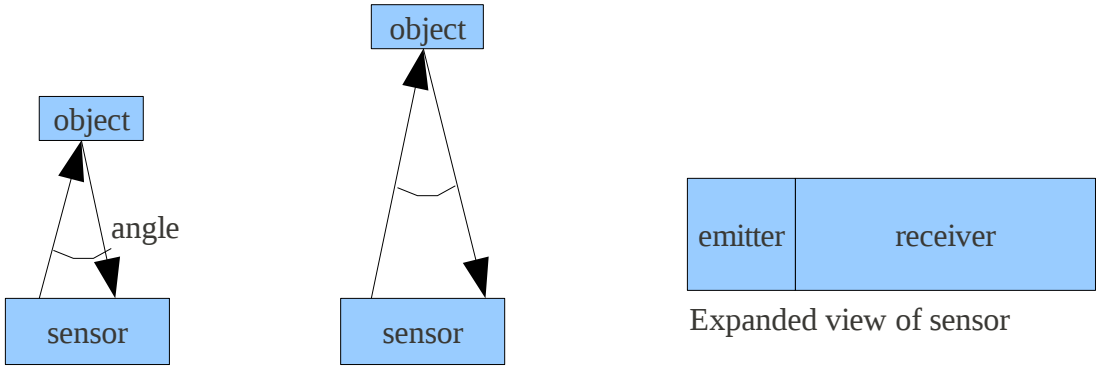
This is the Sharp GP2D12 analogue distance sensor. I am using the Sharp GP2Y0A02YK variant which has a usable range of 20 cm to 150 cm.



Theory of Operation

This distance sensor uses triangulation and a small CCD array to compute the distance to objects in the field of view.

The emitter emits a pulse of IR, if it hits an object it will be reflected back to the CCD array detector. The position at which it hits the detector will depend on how far away the object is

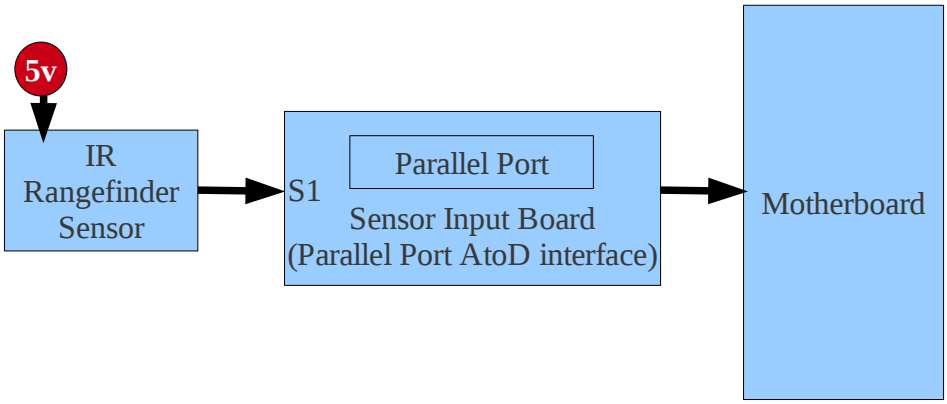


The angles above will vary depending on the distance to the object. The receiver lens transmits the reflected light onto the CCD array based on the angle of the triangle shown above. The CCD array can then calculate the angle of the reflected light and therefore the distance to the object.

Pinout

Pin	Name	Function	Limits
1	Vo	Voltage Output	2.7v @ 20cm <0.5v @ 150cm
2	GND	Ground	-----
3	VCC	Supply Voltage	4.5v to 5.5v
		Supply Current	50mA max.

For testing purposes, I attached the sensor to the Pan & Tilt Head and connected its output to the S1 input of the sensor input board. I also connected it to the 5V supply. The resolution of the sensor board is 1mV per bit. The output of the sensor should vary from about 2.7v at it's minimum distance to 0.4v at the maximum measurable distance



The graph below was taken from the manufacturers data sheet and shows what the sensor output should be at various distances.

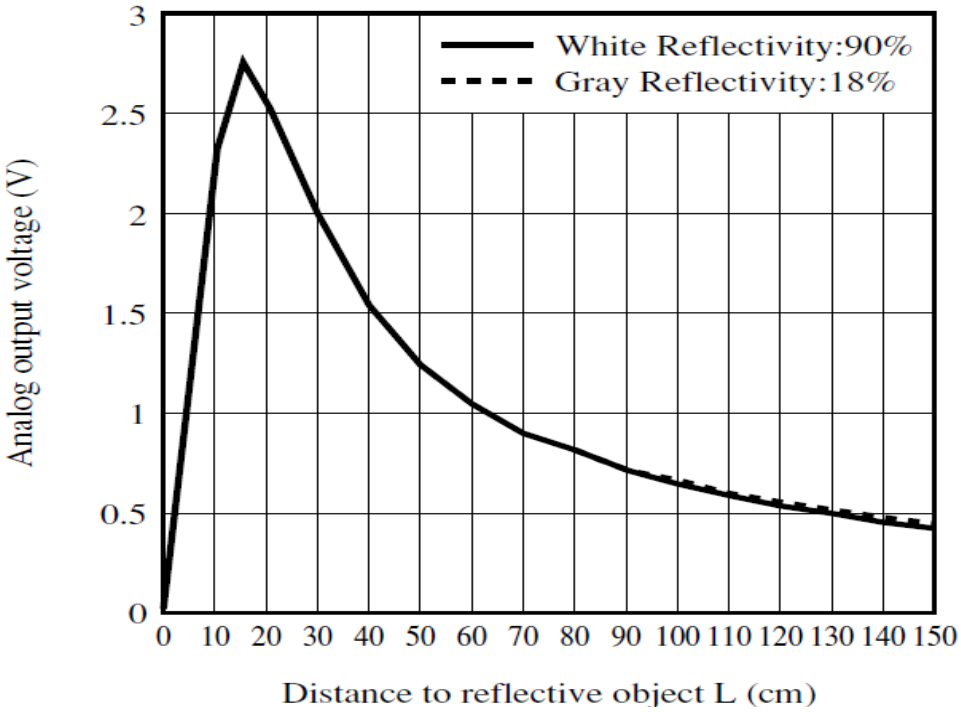
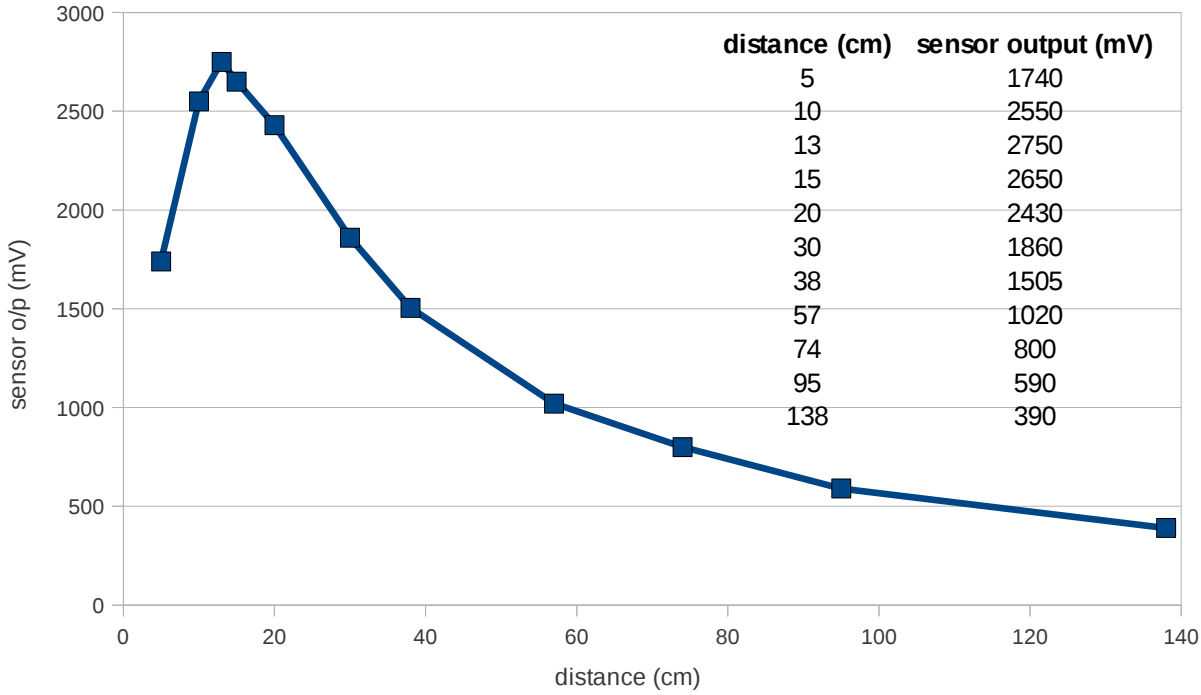


Illustration 1: Manufacturers data - Sensor output vs distance

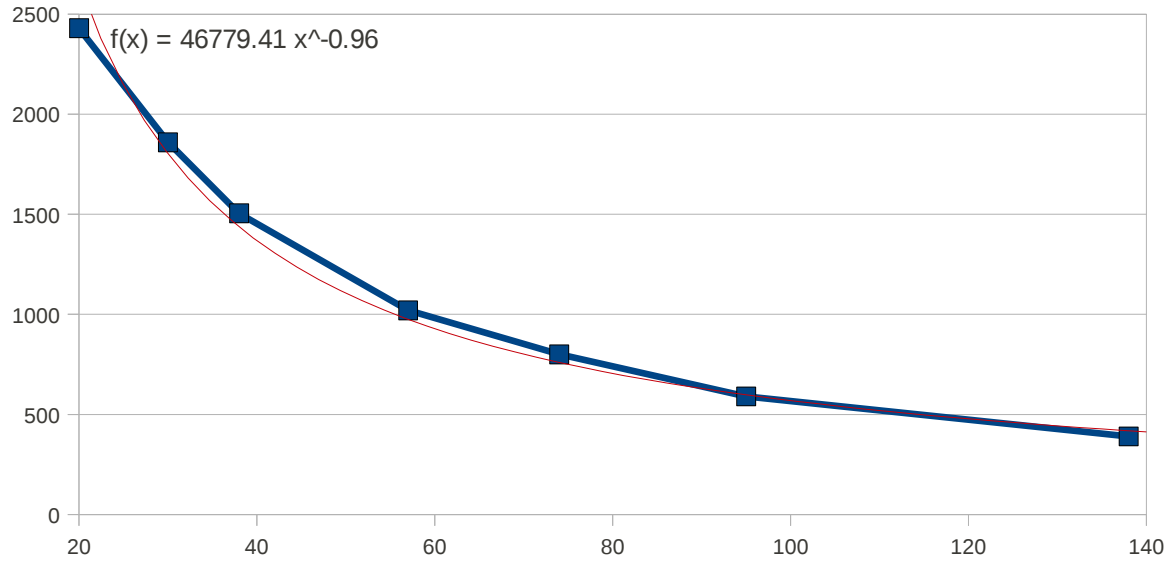
Using a ruler and a plank of wood as the target I took various measurements of sensor output at a range of distances from 5cm to 138cm. This was all very approximate as I used a steel rule & tape measure to measure the distances.



distance (cm)	sensor output (mV)
5	1740
10	2550
13	2750
15	2650
20	2430
30	1860
38	1505
57	1020
74	800
95	590
138	390

This is the actual plot of measured sensor output vs distance. It can be seen from both diagrams above that the sensor output is not linear. It can also be seen that at distances of less than 20cm, the sensor output will start to rise again. This will give rise to a false reading making something close up appear further away than it actually is. I hope to overcome this by using an additional shorter range IR distance sensor (4-30cm range).

If I ignore the results from less than 20cm I now get the graph shown below:



As the sensor output isn't linear I will need the formula for a linear trend line. This was achieved by using OpenOffice.org Calc. By entering the measured values into the spreadsheet I was able to plot the above graph and fit a trend line of type 'Power'. There is an option in 'Calc' to show the formula. The trend line looks like it fits the data adequately for my purpose.

f(x) is the sensor output voltage (mV)
x is the distance (cm)

By using a bit of algebra:

$$x = \frac{46779.41}{f(x)}^{1.0417}$$

The formula works!

So, if if the sensor output is 1020mV then:

Distance = (46779.41/1020)^1.0471 = 55mm. It can be seen from the graph above that this is approximately correct, or good enough for what I want.

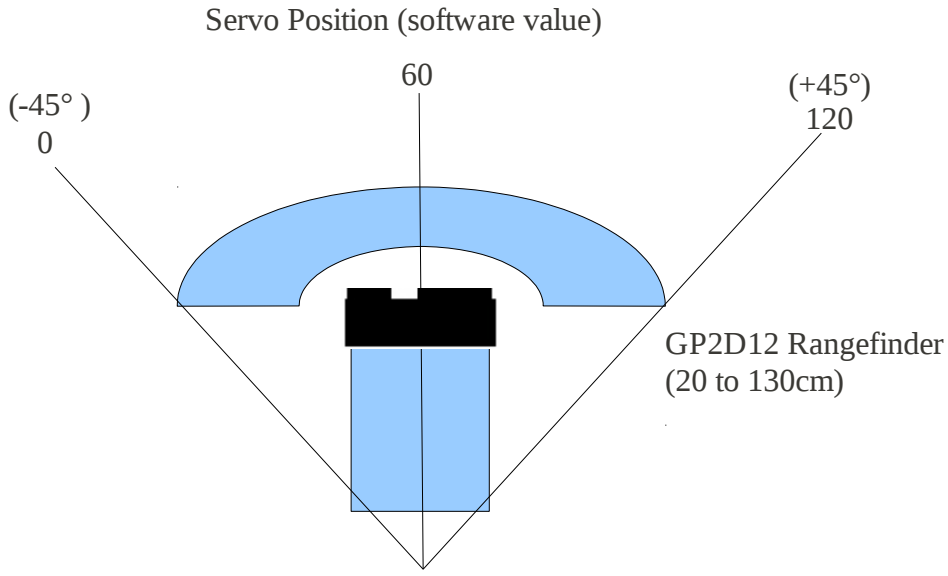
PUTTING IT ALL TOGETHER

Software Design

Simple Algorithm

1. Turn Power On
2. Wait while Robot boots up.
3. Check Battery Voltages
4. Pan head across full field of view and gather sensor data
5. Select direction of movement
6. Move
7. GOTO 3

The steps which requires the most though are step 4 and 5.



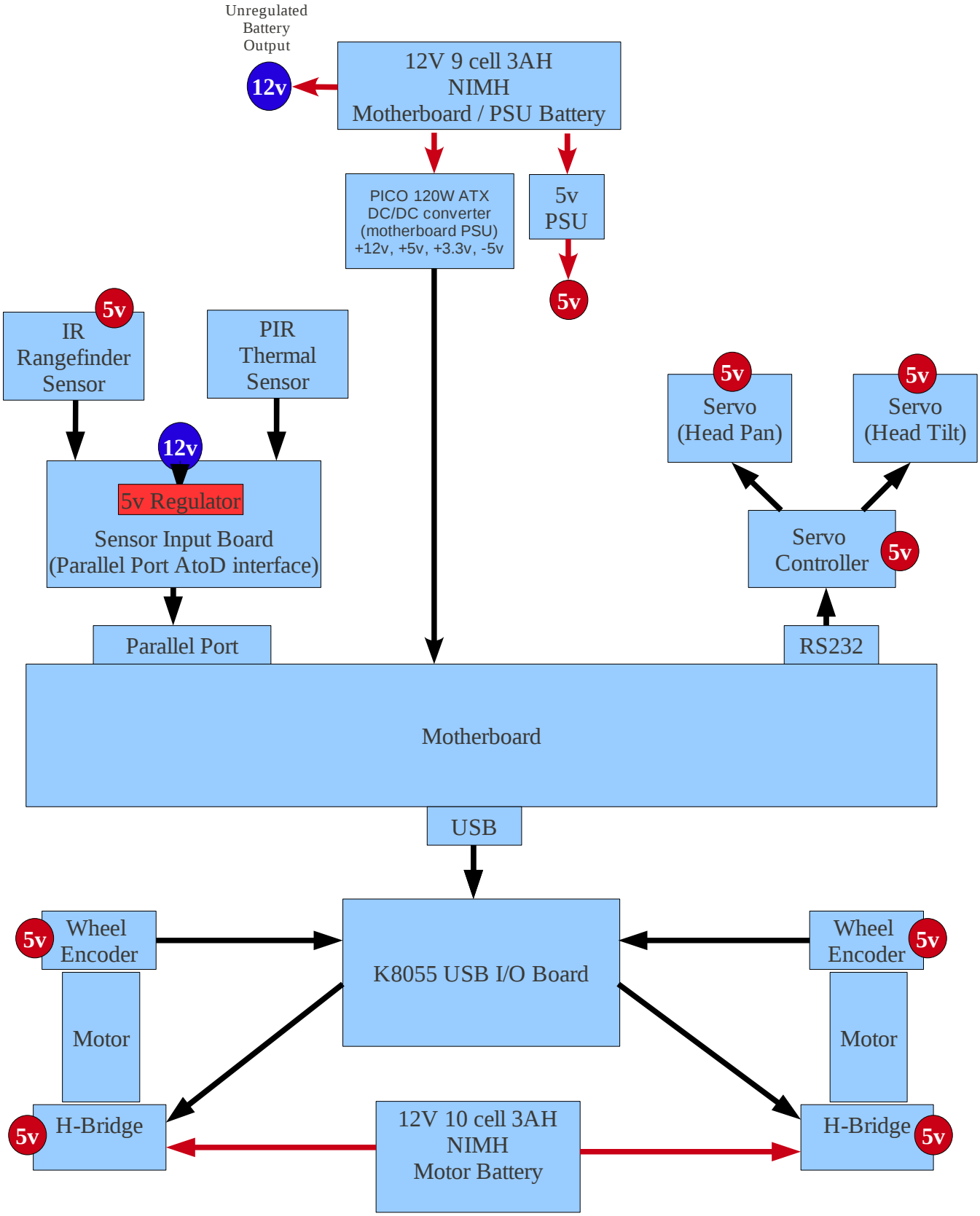
|At each servo position a sensor reading is taken and stored in an array 'scan_data(120) '. The voltage output will vary depending on the distance to any objects within its field of view. The higher the number the closer the object.

Format of Data

Data is in the following format:

Servo position	Sensor output	Servo position	Sensor output
0	464	65	804
5	462	70	1600
10	287	75	1448
15	428	80	1468
20	332	85	1485
25	307	90	1507
30	288	95	1515
35	210	100	1514
40	250	105	1486
45	2248	110	1455
50	2276	115	1434
55	2118	120	1419
60	1861		

APPENDIX 1. SYSTEM



APPENDIX 2. MONITORING BATTERY VOLTAGE

My original plan was to use a 10 cell NIMH battery to power the motherboard via a PicoPsu. Unfortunately, when fully charged this had a no load voltage of about 14.0V. The maximum voltage on the 12V line should be 12.6V according to the ATX specification.

I then experimented with using a 9 Cell NIMH battery to power the motherboard, however the battery voltage seemed to drop fairly quickly. Within 50mins the 12V line of the motherboard fell lower than 11.4V, which is the lowest the 12V line is allowed to drop. To confirm this, I conducted the following measurement.

First, the battery pack was charged for a period of 15 hours at a rate of approximately 0.1C.

	Voltage across battery terminals	Voltage across Motherboard 12V line
Fully charged battery (No Load)	12.9 Volts	XXXXXXXXXXXXXXX
Motherboard booted into Windows	12.6 Volts	12.4 Volts

Time taken for Motherboard 12V line to drop to 11.4V : 50 Minutes.
(The lowest the 12V line should be allowed to drop is 11.4V as stated in the ATX specification.)
This only allows for 50 minutes of use between battery charging.

I went back to the idea of using a 10 cell battery to power the motherboard.
The battery pack was charged for a period of 15 hours.
After allowing the battery to settle for 1 hour I measured the no load voltage, which was ***14.0V.***
The PicoPsu I am using has input voltage protection, which will shut the PSU down if the voltage is above approximately 13.5V. At a voltage of 14V the PicoPsu did indeed shut down. I waited a further 24 hours, during which time the battery voltage dropped to **13.7V and the PicoPsu didn't shut down.**

I then obtained the following measurements:

	Voltage across battery terminals	Voltage across Motherboard 12V line
Fully charged battery (No Load)	13.7 Volts	XXXXXXXXXXXXXXX
Motherboard booted into Windows	13.4 Volts	13.2 Volts

Time taken for Motherboard 12V line to drop to 11.4V : 3 hours.

Although the voltage was 13.2 Volts when the motherboard was first booted, it dropped to 12.6 Volts within 20 minutes.

My 10 cell battery packs are second hand, but have good capacity. A newer battery pack may cause difficulties if the fully charged voltage is much above 14.0V no load, due to the PicoPsu over voltage protection.

APPENDIX 3. MOTOR ENCODER / PROGRAM LISTING

This program uses some cammands that are specific to the Velleman K8055 USB Interface Board, via a supplied DLL.

```
Option Explicit

Rem ** DECLARE VARIABLES **
Rem rpm
Dim rev1 As Integer
Dim rev2 As Integer
Rem PWM amount on each motor
Dim motor1_pwm As Integer
Dim motor2_pwm As Integer
Rem current direction of robot
Rem >> FORWARD, REVERSE, FORWARD LEFT, FORWARD RIGHT, REVERSE LEFT, REVERSE RIGHT  STOP
Dim direction As String
Dim temp As Integer

Private Sub Form_load()
Rem ** AS SOON AS THE PROGRAM LOADS, ATTEMPT TO CONNECT TO THE K8055 USB INTERFACE CARD
Dim CardAddress As Long
Dim h As Long
   CardAddress = 0
   CardAddress = 3 - (Check1(0).Value + Check1(1).Value * 2)
   h = OpenDevice(CardAddress)
   Select Case h
      Case 0, 1, 2, 3
         Label1.Caption = "Card " + Str(h) + " connected"
      Case -1
         Label1.Caption = "Card " + Str(CardAddress) + " not found"
         Beep
   End Select

   Dim response As String
   Rem check to see that the USB card has opened.
   If (h < 0) Then response = MsgBox("Failed to open K8055 USB Interface Card", vbOKOnly, "Can't Continue!")
   If (h < 0) Then
      Rem if the card hasn't opened, then do nothing.
      update_movement.Enabled = False
      Timer1.Enabled = False
      change_dir.Enabled = False
   End If
   Rem ** RESET THE TWO ENCODER COUNTERS ON THE K8055 CARD
   ResetCounter (1)
   ResetCounter (2)
   Rem Set inital level for PWM
   motor1_pwm = 100
   motor2_pwm = 100
   direction = "STOP"
   temp = 0
End Sub

Private Sub change_dir_Timer()
Rem This is for testing only
Beep
temp = temp + 1
If temp = 1 Then direction = "FORWARD"
If temp = 2 Then direction = "STOP"
If temp = 3 Then direction = "REVERSE"
If temp = 4 Then direction = "STOP"
If temp = 5 Then direction = "FORWARD RIGHT"
If temp = 6 Then
   direction = "STOP"
   temp = 0
End If
End Sub

Private Sub read_battery_Timer()
label_v1.Text = ReadAnalogChannel(1) / 10
End Sub

Private Sub update_movement_Timer()
Rem ** THIS CODE IS RUN EVERY 500mS
update_movement.Enabled = False
   Rem show the current direction of movement
   Label_direction.Caption = direction

   Rem calculate rpm for each wheel
   Label_rpm1.Caption = ReadCounter(1) / 20 * 60
   Label_rpm2.Caption = ReadCounter(2) / 20 * 60
   Label_speed1.Caption = ((ReadCounter(1) / 20 * 60) * 0.366) / 60
```

```
    Label_speed2.Caption = ((ReadCounter(2) / 20 * 60) * 0.366) / 60

    If direction = "FORWARD" Then
        Rem ** set H-Bridge 1 input to 0 1 - FORWARD
        ClearDigitalChannel (1)
        SetDigitalChannel (2)
        Rem ** set H-Bridge 2 input to 0 1 - FORWARD
        ClearDigitalChannel (3)
        SetDigitalChannel (4)
        Rem ** set PWM on both H-Bridges.
        OutputAllAnalog motor1_pwm, motor2_pwm

        Rem ** Find out how fast each wheel is turning by reading the two counters (one for each wheel)
        Rem ** on the K8055 board
        rev1 = ReadCounter(1)
        rev2 = ReadCounter(2)
        Rem ** Only vary motor 2 pwm level to maintain same speed as motor 1
        If rev2 < rev1 Then motor2_pwm = motor2_pwm + 2
        If rev2 > rev1 Then motor2_pwm = motor2_pwm - 2
        Rem ** reset the encoder counters
        ResetCounter (1)
        ResetCounter (2)
    End If

    If direction = "REVERSE" Then
        Rem ** set H-Bridge 1 input to 0 1 - FORWARD
        ClearDigitalChannel (2)
        SetDigitalChannel (1)
        Rem ** set H-Bridge 2 input to 0 1 - FORWARD
        ClearDigitalChannel (4)
        SetDigitalChannel (3)
        Rem ** set PWM on both H-Bridges.
        OutputAllAnalog motor1_pwm, motor2_pwm

        Rem ** Find out how fast each wheel is turning by reading the two counters (one for each wheel)
        Rem ** on the K8055 board
        rev1 = ReadCounter(1)
        rev2 = ReadCounter(2)
        Rem ** Only vary motor 2 pwm level to maintain same speed as motor 1
        If rev2 < rev1 Then motor2_pwm = motor2_pwm + 2
        If rev2 > rev1 Then motor2_pwm = motor2_pwm - 2
        Rem ** reset the encoder counters
        ResetCounter (1)
        ResetCounter (2)
    End If

    If direction = "FORWARD RIGHT" Then
        Rem ** set H-Bridge 1 input to 0 1 - FORWARD
        ClearDigitalChannel (1)
        SetDigitalChannel (2)
        Rem ** set H-Bridge 2 input to 0 1 - REVERSE
        ClearDigitalChannel (4)
        SetDigitalChannel (3)
        Rem ** set PWM on both H-Bridges.
        OutputAllAnalog motor1_pwm, motor2_pwm
    End If

    If direction = "STOP" Then
      OutputAllAnalog 0, 0
      ClearAllDigital
    End If


update_movement.Enabled = True
End Sub

Private Sub Command_exit_Click()
    ClearAllAnalog
    CloseDevice
    End
End Sub

Private Sub Form_Terminate()
    ClearAllAnalog
    ClearAllDigital
    CloseDevice
End Sub

Private Sub Timer1_Timer()
    Timer1.Enabled = False
    Dim i As Long
    Rem Dim Data1 As Long
```

```
    Rem Dim Data2 As Long
    Rem read in the states of digital inputs I1 and I2
    i = ReadAllDigital
    Check2(0).Value = (i And 1)
    Check2(1).Value = (i And 2) / 2

    Rem Display how many pulses have been counted on inputs I1 & I2
    Label_enc1.Caption = ReadCounter(1)
    Label_enc2.Caption = ReadCounter(2)
    Rem display the PWM level for each motor
    Label_pwm1.Caption = motor1_pwm
    Label_pwm2.Caption = motor2_pwm
    Timer1.Enabled = True
End Sub
```

APPENDIX 4. SENSOR INPUT PROGRAM

This program uses some commands that are specific to INPOUT32.DLL. This dll enables program to talk directly to the parallel port of the motherboard. More information can be obtained by searching for it over the internet.

```
Option Explicit
Dim Value As Integer
Dim input_select As Integer
Dim atod_output As Integer
Dim elapsed As Long
Dim readings As Long

Private Sub Command_go_Click(Index As Integer)
' Take a reading
  Call select_input_channel(Index)
' Get data from AtoD converter
  Call get_data(Index, atod_output)
  text_sensor(Index).Text = atod_output
End Sub

********************************************************************************
'Inp and Out declarations for port I/O using inpout32.dll.
Public Declare Function Inp Lib "inpout32.dll" Alias "Inp32" _
    (ByVal PortAddress As Integer) _   As Integer
Public Declare Sub Out Lib "inpout32.dll" Alias "Out32" _
    (ByVal PortAddress As Integer, _  ByVal Value As Integer)

Function BitStatus(PortNum%, BitYouWant%) As Integer
    NumOfBits% = 8
    ReDim PortBits(NumOfBits%) As Integer
    For i = 1 To NumOfBits%
        PortBits%(i) = PortNum% Mod 2
        PortNum% = Fix(PortNum% / 2)
    Next i
    BitStatus = PortBits%(BitYouWant%)
End Function

Function user_delay(delay) As Integer
    Dim n As Single
    n = Timer
    While Timer - n < delay: Wend
End Function

Function get_data(input_select, atod_output) As Integer
Dim status_total As Integer
Dim PortAddress As Integer
'Change PortAddress to match the port address to write to:
'(Usual parallel-port addresses are &h378, &h278, &h3BC)
PortAddress = &H3BC
' put tick in relay on/off box
atod_output = 0
Out PortAddress, 32 + 8 + input_select 'CS High  SCLK Low -
Out PortAddress, 32 + 0 + input_select  'CS Low   SCLK Low - Start a conversion
' Get bit of data
For n = 11 To 0 Step -1
sensors.Check1(n).Value = 0
    ' Clock SCLK - Get bit of data
    Out PortAddress, 32 + 16 + input_select  'CS Low  SCLK High
    Out PortAddress, 32 + 0 + input_select 'CS Low  SCLK Low
    ' Read Status Port
    status_total = Inp(PortAddress + 1)
    status_total = 255 - status_total
    ' Get status of b3 (status port) - 1 or 0.
    status_total = BitStatus(status_total, 4)
    If status_total = 1 Then
        atod_output = atod_output + (2 ^ n)
        sensors.Check1(n).Value = 1
    End If

Next n
atod_output = 4095 - atod_output
DoEvents
End Function

Function select_input_channel(input_select) As Integer
PortAddress = &H3BC
' First select input channel
Out PortAddress, input_select
' Small delay while Multiplexor switches input channels
Call user_delay(1)
```

End Function

TO DO

MENTION BATTERY CONNECTORS USED.

DESIGN SOFTWARE

- MOTOR DRIVE
- HEAD MOVEMENT
- SENSOR INPUT

ATTACH SPEAKER FOR SOUND

- MAYBE NEED AMP.