

The Hovercraft has three main boards. This section provides a detailed overview of each board.

Main Computer Board

The main computer board is the heart of the robot. The main computer board has three main functions:

- 1) Use rangefinder to measure distances to obstacles
- 2) Communicate with ANN board
- 3) Communicate with speed control

The Rangefinder

The main computer board controls “the rangefinder”. The rangefinder is a module that consists of a servo, an ultrasonic transducer and the driver board. The driver board and ultrasonic transducer were both removed from an old Polaroid “Sonar OneStep Land Camera”. This module was later fried, and was replaced by a similar transducer from a “Spectra Sonar” camera, again made by Polaroid. This camera used an ultrasonic transducer to measure the distance to an object for the auto focus of the camera. The driver board could be easily modified to work outside the camera. The board can require up to two amps of current for it to work properly.

The first problem was not too hard to solve. The LM7805 regulator only provides 1.5 amps of current at once. In order to supplement the power, a large capacitor was placed in the power supply lines near the rangefinder board. A power transistor switches the voltage over to the rangefinder board.

A counter counts an arbitrary number that starts when the pulse is sent and stops when the echo is received. This counter is very precise, although its accuracy doesn't matter. The counter needs to always count the same value for an obstacle the same distance away, but this value doesn't actually count the exact time it takes for the pulse to be sent and the echo received. This is because it is simply calibrated: an obstacle's distance is measured and the counter's value recorded. With a few measurements, the relationship between the counter's value and the distance to obstacle can be easily determined.

Finally, the ultrasonic transducer was mounted on a servo. The servo allows the transducer to look at different directions, to try and discover which direction is clear. The servo was a standard airplane servo, with approximately 180 degrees of movement. On the hovercraft however, this is restricted because the transducer starts to impact other parts of the hovercraft when it is rotated too far. The servo is controlled by a Pulse Width Modulated (PWM) signal, that varies from 1 ms to 1.5 ms in pulse width.

Communicate with ANN board

There are two parts to any communications: the physical part and the protocol. The physical part is the medium – for instance writing or speaking. The protocol is how it communicates – such as the language. Even if two people are using the same physical means, such as speaking, the language has to be the same. This holds true for computers as well. With this basic knowledge of what protocols are, let's look at how the main computer board communicates with the ANN board.

The main computer and ANN board use a serial link operating at 9600 baud rate. This is asynchronous communications. In asynchronous communications, the two ends are timed to operate together. For instance a single bit might be 1.5 ms long. This means that if the receiving end saw a low that lasted 3 ms, that is equal to “00”. This is physically how the main computer board and ANN board communicate.

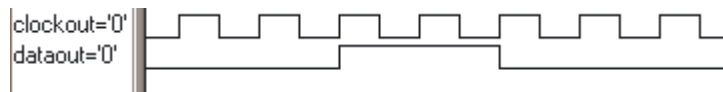
There were two different protocols used for communications. The first one used was simple serial data. The sending end would just send a few data bytes, and wait for a few bytes back. In this case, the main computer board sent the distance to the nearest obstacle, and waited for the ANN to return a number that indicated the course of action it should take. This has some severe problems. Both sides are just blindly sending and receiving data, without any idea if the data was corrupted en route or if it is even the data it is supposed to be getting! Often data DID get corrupted between communications, which was not acceptable. To solve this problem a different protocol was implemented: SNAP.

SNAP stands for Scalable Node Address Protocol. The SNAP protocol is much more reliable. Not only does it define who each packet is to and from, but also allows an Error Detection Method (EDM). EDM results in a huge increase in reliability because it allows the receiving end to detect if any data was corrupted when the packet was en route. This is accomplished by the sending end performing an algorithm on the data, which generates an output number. This output number is added onto the end of the packet, and sent along with the rest of the data. The receiving end takes the data and performs the same algorithm. If the receiving end received the same data, it should get the same output from the algorithm as the sending end. If they differ, then corruption occurred and the receiving end would request that the data be resent. In this project the EDM used was Cyclic Redundancy Check (CRC), with an 8 bit output (CRC8).

Communications with the Speed Control

Three different types of communications between the speed control and the main computer board were used in developing the hovercraft. The first attempt used a very simple Pulse Width Modulation (PWM) scheme to communicate. The main computer board would output a pulse ranging from 1 ms to 1.5 ms in length, which would tell the speed control what speed to set the motors to. However there was a problem: the speed control used an Atmel AVR microcontroller that had an internal RC oscillator, as opposed to a crystal oscillator. This internal oscillator wasn't accurate enough to provide a timebase for measuring the signals properly, so a synchronous method of communications had to be used. In synchronous communications there are two wires: data and clock. The clock synchronizes the sending and receiving end and it can work in many different ways. For example: every time the "clock" line goes high, the data on the "data" line is read. So if the sending end wanted to send the sequence "00110001", it would do the following:

Set data line low
Pulse clock line
Set data line low
Pulse clock line
Set data line high
Pulse clock line
Set data line high
Pulse clock line
Set data line low
Pulse clock line
Set data line low
Pulse clock line
Set data line low
Pulse clock line
Set data line low
Pulse clock line
Set data line high
Pulse clock line



Again the protocol used was the SNAP protocol. If you are interested in more technical details of SNAP, there is some information on SNAP in this binder.

Later once the AT90S2343 was replaced with the AT90S2313, this problem went away. The AT90S2313 can run off an external crystal, which is accurate enough to run the asynchronous serial link. The speed control now uses that serial link and the SNAP protocol. The details of the serial link are very similar to how the main computer board communicates with the ANN board.

Artificial Neural Network (ANN) Board

If the main computer board is the heart, then the ANN board is the brain. The ANN board, which houses the neural network, has two main purposes:

- 1) Communicate with main computer board
- 2) Run Artificial Neural Network

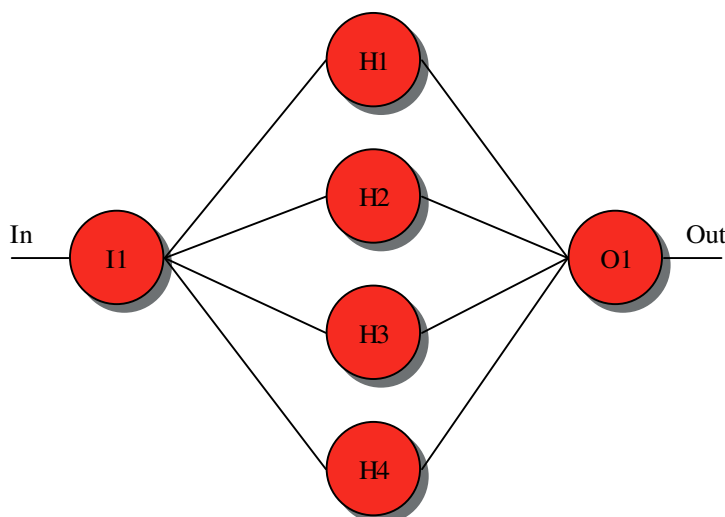
Communicate with main computer board

The specifics of the communications system used were discussed in the main computer board section. However, this is a quick overview of the data sent back and forth. The computer board sends the distance to an obstacle in centimetres to the ANN board. This is stored as an integer on the main computer board that is broken into two bytes, sent over the link, and then reconstructed back on the ANN board to form an integer again. The neural network only accepts inputs in the range of 0 to 1, so the distance had to be converted from a range of 0 to 500 to a range of 0 to 1 (note: this is not a binary input, but any decimal number from 0 to 1, such as 0.2839173). The neural network then returns two numbers, both in the range of 0 to 1. One number is the amount the hovercraft should turn by; the other is the speed the hovercraft should be set to. Both of these numbers are multiplied by 1000, to eliminate the decimals. Each integer is then broken into two bytes (four bytes in all) and sent back to the main computer board. This is how the neural network receives and transmits data.

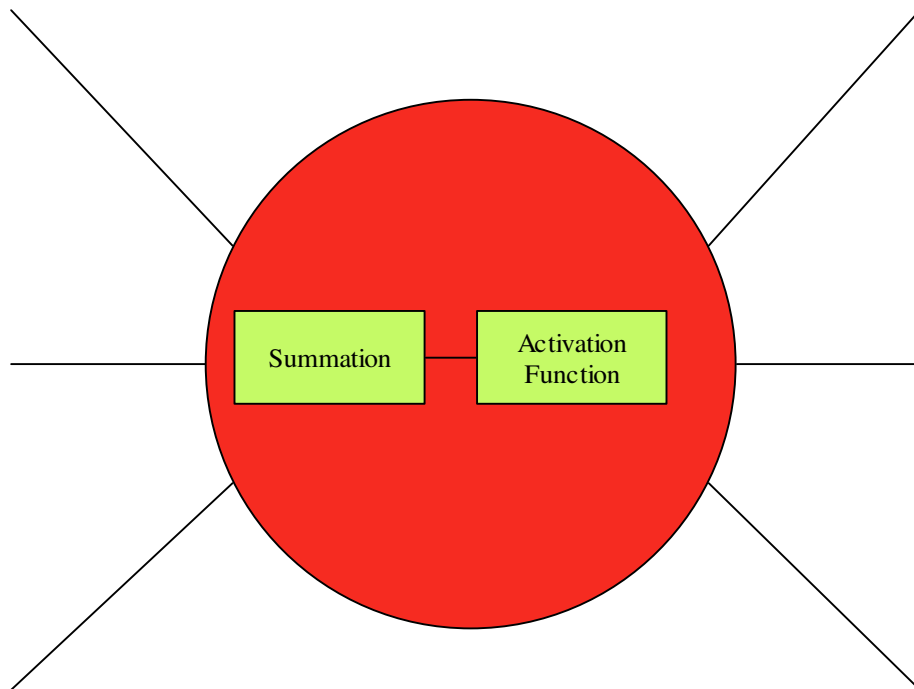
The Artificial Neural Network

Note: There is another more detailed section of this binder that talks about the specific activation functions of neural networks, as well as shows the exact training and weight files used in this hovercraft.

The artificial neural network is the key to this project. A neural network uses the interconnection of neurons to solve a problem. Here is a simple model of a neural network:



This simple neural network has one input neuron (I1) in the input layer, four hidden neurons (H1 – H4) in the hidden layer, and one output neuron (O1) in the output layer. Then each connection is “weighted” between neurons. So, for example, the neuron O1 will come up with its output based on the input it gets from H1, H2, H3 and H4. The output is the sum of the weighted connections from the other neurons. Let’s look “inside” an artificial neuron to see what it looks like.



The weighted inputs are first summed together, then passed through the activation function. Before this happens, each input is multiplied by its “weight”. In this way a neural network is “trained”, not programmed. There are many different methods for training a neural network, as this field is always growing and changing. The method used by the training software that generated the weight files on the ANN board is called “back propagation” and works like this:

- 1) Feed the input through the neural network, just like you would when normally running data through a neural network.
- 2) Find the difference between the output and the desired output (this is the error), and locate the neuron responsible for this error. Adjust the weighted connections to minimize this error.

- 3) Repeat until the error is gone (or small enough to not be of concern).

When the back propagation goes through the first time, all the weighted connections are completely random. This means the error is huge! However, it soon starts to reduce itself until it is too small to worry about. Sometimes the error never goes away though. When this happens, it means there is a problem with the neural network. Usually increasing the number of neurons on the hidden layer or increasing the number of hidden layers (there can be more than one) will solve this. However a neural network can quickly get too large to manage. A neural network with 2 neurons in the input layer, 8 neurons in the hidden layer and one neural in the output layer has 24 weighted connections. The weight of each connection must be assigned. However a neural network with 80 neurons in the input layer, 300 neurons in the hidden layer and 10 neurons in the output layer would have 27 000 weighted connections to be trained! This is still a tiny fraction of the number in the human brain! The output from the training software is called the “weight file”, and is a list of numbers, each number being the weight for one connection of a neuron to a neuron.

There are many software programs that run on PC's to generate weight files. For this project software called “Neural Lab Version 2.5” was used.. This weight file is then programmed into the FLASH memory of the AVR microcontroller, along with the program. The program starts off by taking this weight file and loading it onto the SRAM. The SRAM provides a way for the microcontroller to quickly and easily access the weight file. Once the weight file is loaded into RAM, the processor gets the distance to the nearest obstacle. It then goes through every connection to every neuron and applies this weight to each connection. The activation function is also applied to each neuron. The two neurons on the output layer's value are read, and this value is sent back to the main computer board.

As you can see, this ANN board truly is the brain of the robot. A brief overview of neural networks has been presented here, however the subject is very complicated, and there are large books written on the subject of neural networks.

Again, there is a section in this binder that contains more detail on the specific neural networks, as well as the training file used by the neural network in the hovercraft.

Speed Control

The speed control is a fairly simple board. It controls the speed of the propulsion and lift motors. The speed control board only has two purposes:

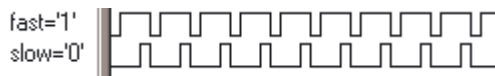
- 1) Communicate with the main computer board
- 2) Control speed of both motors

Communications with the main computer board were fairly simple and the specifics were discussed in the “Main Computer Board” section. The main computer board sent out two bytes: one defines the speed to set the propulsion motor to, and the

other was a “panic mode” byte. If the “panic mode” byte was set, the main computer board has decided that there is nowhere to go. When this happens, the speed control turns both the lift fan and propulsion motor off which will stop the hovercraft. Otherwise, the lift fan is set to the same speed, and the propulsion motor is set to the proper speed.

Control Speed

The speed of both motors is controlled by Pulse Width Modulation (PWM). PWM speed control works not by varying the voltage that each motor receives, but varying how long it receives this voltage for. Take a look at this timing diagram. The top trace shows



that the motor would receive 12 volts for much longer than it is off. Since the periods of these on-off cycles are very small, the motor does not pulsate but instead is simply an average. In the case of the top trace where the motor is on 75% of the time, the speed would be approximately 75% of maximum speed (not exactly, but somewhat close). In the bottom trace the motor is off for longer than it is on. It is only on about 25% of the time, which means that the motor will be slower than the above trace.

The motor’s power is switched on and off by a power MOSFET (Metal Oxide Semiconductor Field Effect Transistor). Power MOSFETs are capable of dealing with large amounts of current, which is perfect for this speed control. Originally, two power MOSFETs were used in parallel for the propulsion motor, and one power MOSFET used for the lift motor. However, the lift motor’s MOSFET blew out, and was replaced by one of the MOSFET’s from the propulsion motor circuit. The MOSFETs for the propulsion provided a large margin for stall-current, so the single power MOSFET should be able to power the propulsion motor safely.

The power MOSFET has three terminals called Drain, Source and Gate. When the gate is off, Drain and Source are in a high-resistance state. In other words, they are off. When a voltage is applied to the gate pin, the drain and source slowly go from a high-resistance state to a low-resistance state. For the drain and source to have the lowest possible resistance the gate voltage has to be fairly high, anywhere from 10 to 15 volts for a normal power MOSFET. The problem is that if the voltage is below that required to go into a low resistance state, the power MOSFET will be in a middle-resistance state. Here the resistance could be at a value low enough to pass significant current, but high enough that a lot of heat is generated. For instance, if the resistance from source to drain was 1 ohm, then the power MOSFET would be dissipating 60 watts of power if the voltage was 12 volts and the load was drawing 5 amps. However, at a source to drain resistance of 0.024 ohms the same 12-volt, 5-amp load would make the power MOSFET dissipate a tiny 0.288 watts (in theory)! This is why the gate drive circuitry is important. This speed control takes advantage of “logic level” MOSFETs. They are designed to be switched fully on by standard TTL output of a chip (5 volts). This greatly simplifies the design, as the MOSFET can be connected directly to the output of the microcontroller.

There is also another design consideration. It is impossible to go from 0 volts to 5 volts on the gate without passing all the in-between voltages. Those in-between voltages

make the MOSFET go into the state where it will dissipate a lot of heat. Every time the MOSFET goes from 0 to 5 volts (turn MOSFET 'on') or 5 to 0 volts (turn MOSFET 'off') on the gate pin it will go through a transition. During this transition the MOSFET will heat up. It is important to minimize these transitions when possible. For example if you switch the MOSFET controlling the motor on and off 200 000 times a second, you go through 400 000 of these transitions, and the MOSFET will heat up a tiny bit during each one. However if you only switch the MOSFET controlling the motor on and off 200 times a second, that is only 400 transitions. Since you don't have as many transitions, the MOSFET will not get as hot. There are other problems as well though. If you switch the MOSFET controlling the motor at an audible frequency (20 Hz to 20 kHz), then the motor might actually emit an audible whine as it is switched on and off. So for human comfort reasons it is desirable to switch above 20 kHz to avoid a high-pitch whine from the motor.