# TRIBHUVAN UNIVERSITY
# INSTITUTE OF ENGINEERING

Kathmandu Engineering College
Department of Electronics and Communication Engineering

Final Year Project Report

On

# ROBOTIC VACUUM CLEANER

[Code No: EX755]

By
Abhishek Koirala - 69053
Adarsh Bhattarai - 69055
Amrit Khatiwada - 69058
Bijay Dulal - 69073

Kathmandu, Nepal

2071

# TRIBHUVAN UNIVERSITY
# INSTITUTE OF ENGINEERING

Kathmandu Engineering College
Department of Electronics and Communication Engineering

Final Year Project Report

On

# ROBOTIC VACUUM CLEANER

[Code No: EX755]

By

Abhishek Koirala - 69053
Adarsh Bhattarai - 69055
Amrit Khatiwada - 69058
Bijay Dulal - 69073

Kathmandu, Nepal

2071

**TRIBHUVAN UNIVERSITY**
**INSTITUTE OF ENGINEERING**

Kathmandu Engineering College
Department of Electronics and Communication Engineering

# ROBOTIC VACUUM CLEANER

[Code No: EX755]

PROJECT REPORT SUBMITTED TO
THE DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE BACHELOR OF ENGINEERING


By

Abhishek Koirala - 69053
Adarsh Bhattarai - 69055
Amrit Khatiwada - 69058
Bijay Dulal - 69073

Kathmandu, Nepal
BHADRA 2071
TRIBHUVAN UNIVERSITY
Kathmandu Engineering College
Department of Electronics and communication Engineering

# TRIBHUVAN UNIVERSITY
# INSTITUTE OF ENGINEERING

Kathmandu Engineering College
Department of Electronics and Communication Engineering

## CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Electronics and Communication Engineering, a final year project work entitled "ROBOTIC VACUUM CLEANER" submitted by Abhishek Koirala, Adarsh Bhattarai, Amrit Khatiwada, Bijay Dulal in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

Er. Sarbagya Ratna Shakya
(Project Supervisor)
Department of Electronics and Communication Engineering.
Kathmandu Engineering College

Dr. Ram Krishna Maharjan
(Associate Professor)
Electronics and Computer Engineering Department,
Institute of Engineering,
Tribhuvan University,
Kathmandu, Nepal

Er. Sarbagya Ratna Shakya
(Project Coordinator)
Department of Electronics and Communication Engineering.
Kathmandu Engineering College

Assoc Proff. Shiva Raj Baral
(Head of Department)
Department of Electronics and Communication Engineering.
Kathmandu Engineering College

## ACKNOWLEDGEMENT

It is our pleasure to present this project report on “Robotic Vacuum Cleaner”.

We must express our sincere gratitude towards Kathmandu Engineering College and Department of Electronics and Communication for providing us this opportunity to work towards the completion of our project entitled “Robotic Vacuum Cleaner”. We also acknowledge the College management for providing us with the necessary resources and coordination of the college staff. Finally we express our gratitude to our Project Coordinator and Supervisor Mr. Sarbagya Ratna Shakya for providing us with his valuable guidance and assistance during the whole cycle of the project development.

At last, we thank all our teachers, friends and all other staffs who helped us in successful completion of this project

Thank You.

Project members:

Abhishek Koirala

Adarsh Bhattarai

Amrit Khatiwada

Bijay Dulal

## ABSTRACT

Autonomous and Semi-autonomous platform are the key aspects of today's technological world. The technology has also been applied to different needs of the human beings, automobiles and so on to make the life of human beings much easier employing various tools, mechanism and resources. With our project "Robotic Vacuum Cleaner", we intend to develop a robot capable of vacuuming the floor of a room or area thus saving the valuable human time.

This robot comprises of three basic modules, controller part, robot and a vacuum cleaner. The vacuum cleaner and obstacle detector sensors will be integrated and controlled by a remote which is useful in cleaning our home, industries and with little modifications can be used in agriculture field.

The documentation of this project includes all the details designed during the analysis and designing phase as well as the implementation phase. Hence this report gives the clear view of the process of the development of the project.

# Table of Contents

# LIST OF FIGURES

# LIST OF TABLES

# Acronyms

| | |
|---|---|
| AC | Alternate Current |
| ASCII | American Standard Code for Information Exchange |
| ASK | Amplitude Shift Keying |
| BPS | Bits per Second |
| C | Capacitor |
| CLK | Clock |
| CMOS | Complementary Metal Oxide Semiconductor |
| DC | Direct Current |
| EEPROM | Electrically Erasable Programmable Read Only Memory |
| FSK | Frequency Shift Keying |
| GND | Ground |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| I/O | Input Output |
| IC | Integrated Circuit |
| ICSP | In-Circuit Serial Programming |
| IP | Internet Protocol |
| IR | Infrared |

| | |
|---|---|
| KB | Kilo-Byte |
| mA | Milli-Ampere |
| PC | Personal Computer |
| PCB | Printed Circuit Board |
| PWM | Pulse Width Modulation |
| R | Resistor |
| RAM | Random Access Memory |
| RD | Read |
| RF | Radio Frequency |
| RX | Receiver |
| SRAM | Static Random Access Memory |
| TX | Transmitter |
| USB | Universal Serial Bus |
| V | Voltage |
| VB | Visual Basics |
| Vcc | Positive Power Supply |
| VCR | Video Cassette Recorder |
| QPSK | Quadrature Phase Keying |

# CHAPTER 1
# INTRODUCTION

## 1.1 OVERVIEW

Robot is an electromechanical device which automates to work in many areas like industrial sectors, military application, domestic works, agriculture applications, etc. Robots are reliable especially in areas where human interventions is rather impossible or can cause hazardous effect on human health.

The field of robotics has matured over the last few decades, the technologies have become cheaper and more widely available for application on new robotic projects. An important application for robotic automation to everyday life is the development of intelligent robots which intend to make human life easier.

## 1.2 PROJECT INTRODUCTION

This project has been aimed on developing a remote controlled vacuum cleaner robot with an automatic mode included, based on the principles of robotics. In this project, we have built a vehicular robot and attached a vacuum cleaner with it. The communication between the PC and the car is based on Radio Frequency such that the direction to the car is given through computer's keyboard.

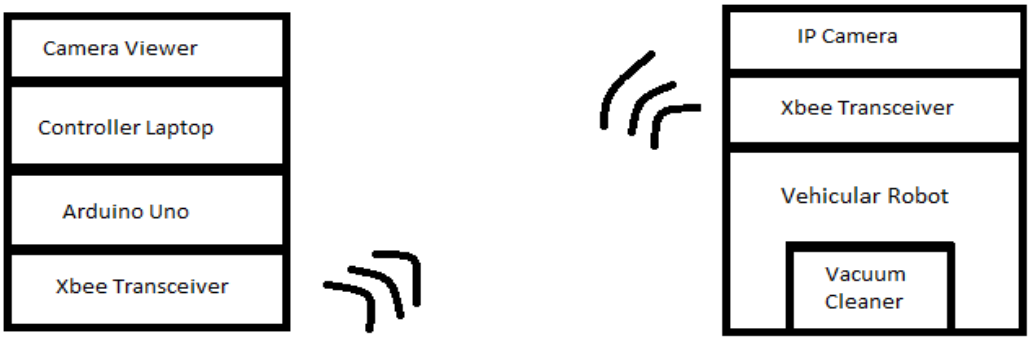



Figure 1.2 Physical Architecture of Robotic Vacuum Cleaner

## 1.3 ORIGIN OF IDEA

The idea of implementing a robotic vacuum cleaner was inspired by our success in one of our projects which implemented similar concepts for wirelessly controlling a robotic car. Though we completed that project, it really had very less applications so we decided to add a vacuum cleaner in the car so that its application range could be widened. Also we were very much interested in adding sensor based movements in our vehicle so that it would not require manual inspection every time it runs.

## 1.4 PROBLEM STATEMENT

Problem of moving a robot through known and unknown environment has attracted much interest over past few decades. Such problems have several difficulties and complexities that are unobserved. Since a robot may encounter obstacles of all forms that must be bypassed in an intelligent manner, this project was aimed at developing a robotic vehicle that would be able to detect obstacles and edges as well as clean the surface in its paths either in manual control mode or automatic control mode in addition to providing video feedback of the surrounding around the robot.

## 1.5 CHARACTERISTICS OF THE PROJECT

- Edge detection with the use of TCRT.
- Vacuum Cleaner for cleaning purpose
- Sonar sensor for obstacle detection.

## 1.6 OBJECTIVES

The main objective of this project was to develop a way to interface a vehicular robot with the computer wirelessly. In addition to that our objective was to embed the vehicle with a vacuum cleaner so that it could clean the surface in its path of movement. Our target was to implement sensors so that the robot could detect obstacles as well as edges and avoid them.

### 1.6.1 PRIMARY OBJECTIVES

1. Interface a vehicular robot with computer or remote wirelessly.
2. Attach a Vacuum Cleaner in order to clean the desired location.
3. Implement sensor based movement to avoid obstacles and edges.

### 1.6.2 SECONDARY OBJECTIVES

1. We also wanted to learn about the autonomous and semi-autonomous concept which has been a topic of concern in the robotics field and be familiar with it
2. We wanted to get some practical knowledge about the application of various electronics components we had been learning about.
3. We wanted to be familiarized with Serial Data communication and the practical application of Radio-Frequency.

## 1.7 COMPONENTS USED

### 1.7.1 HARDWARE COMPONENTS

1. Arduino Uno
2. XBEE Transceiver
3. L293D
4. DC Motors
5. Vacuum Cleaner
6. IP Camera
7. Sonar
8. TCRT
9. ADSL router

### 1.7.2 SOFTWARE COMPONENTS

1. Microsoft Visual Basic
2. Arduino Programming

# CHAPTER 2
# LITERATURE REVIEW

The idea of robot performing the work of basic human work had been in imagination from early times. As time advanced and imagination began to flow around people mind, the concept of semi-autonomous came in play. When technology was still focused on semi-autonomous robots, the science took itself to autonomous platform in movies, literature, science articles, and opened whole new possibilities for humans to explore. Soon the robots were vital to us for deep-sea exploration, hazardous waste-site, nuclear reactors, deep space exploration and so on. With the completion of our semi-autonomous platform in our minor project, we were focused to move on autonomous platform which we have successfully implemented in our major project.

The autonomous projects have been very few in Engineering. Our project "Robotic Vacuum Cleaner" combines the idea of autonomous with semi-autonomous platform included, and realization of the platform such as "obstacle detection" and "edge detection" in a single project. As an additional feature, we have also added camera to receive video feedback of the robot from a remote place.

The implementation of robot avoiding obstacle and edges without the help of manual control was successfully completed before the manual control phase. To make certain that robot can be controlled via a controller in case of need, we implemented the manual control via Xbee and PC keyboard.

While searching for the past projects related to robotic vacuum cleaner, we came across projects such as "Autonomous Robotic Vacuum Cleaner" developed by Eric So, Joaquin Ajtum, Ying Moy and Yueh Lan Quachand and "ROBOTIC VACUUM CLEANER" developed by Hiba Ghannam and Hawa' Osama.

In the first project, the data were taken from an array of inputs that told the condition of the floor space around the vacuum, the sensors used were sonar, touch sensors, and a digital compass. The data from these inputs were fed into the chip(s) which through its software program decided which direction the vacuum should move by sending the control signals out to the drive motors. The rotating brushes were used to pick up the

particles swept into the cleaning path. In the second project, the robot was constructed to avoid any obstacle at any corner of a given area, discrete or continuous, using different algorithms and C programming language. IR sensor and ultrasonic sensors were used in this project.

These projects further helped us to develop our robot capable of cleaning the floor, and to add new features on our project such as edge detection to avoid from falling off cliffs, in both manual and automatic configuration.

# CHAPTER 3
# DESCRIPTION OF PROJECT

## 3.1 BASIC BLOCK DIAGRAM

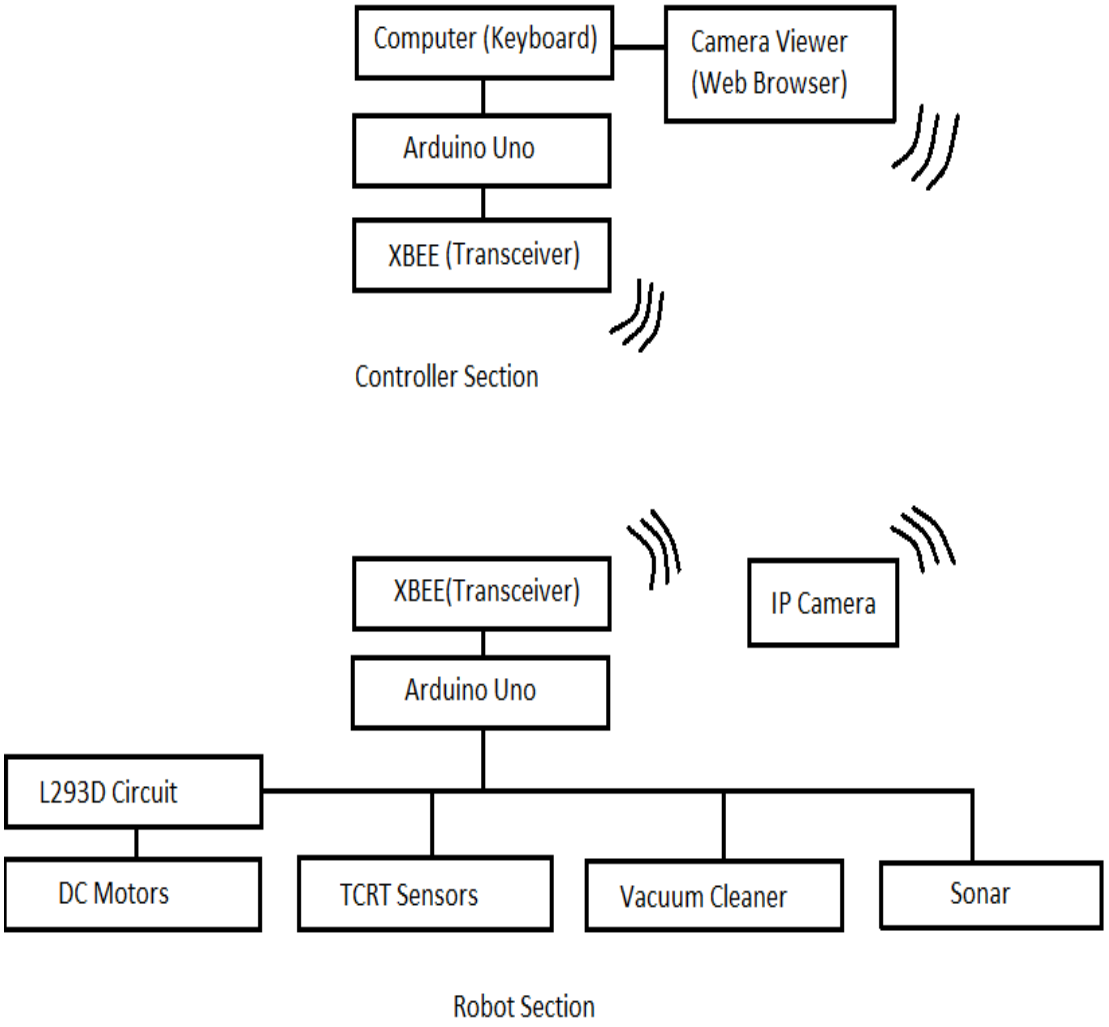



Figure 3.1 Basic Block Diagram of Robotic Vacuum Cleaner

## 3.2 HARDWARE DESCRIPTION

### 3.2.1 XBEE TRANSCEIVER

Xbees are a form of wireless communication. They use radio frequency (RF) to communicate over long distances (from 100m to over 1500m.) They do have issues transmitting through walls and through obstacles, but that is common to many types of wireless communication.There are two main types of Xbees: Series 1 and Series 2. In our project, we have used Xbee pro series 2.The series 2 consumes lower current, with data rate around 20kbps and with coverage area up to 100m (indoor) and 1km (outdoor). We have also used Xbee XSC shield module to connect the Xbee transceiver with Arduino Uno.The modulation used is QPSK,Quadrature Phase Shift Keying.

Before getting Xbees to communicate with each other, it may be necessary to configure. This tells the Xbees which channel they will send or receive data from and which Xbee they are communicating with. Configuring can be done easily with the use of a software called X-CTU. The Xbee shield module was vital to configure the two Xbees via X-CTU. We had to set Arduino Uno in reset mode to be able to configure the Xbee in X-CTU by short-circuiting reset pin and GND pin in Arduino Uno. We configured one Xbee as Coordinator AT and other as Router AT. We set the same PAN ID for both Xbee so the two Xbee could communicate with each other.

Xbee transceiver module consists of 20 pins.The pin diagram of xbee transceiver with detailed description and their functionality along with the electrical specification of xbee are given in next page.

| Pin | Nam | Direction | Description | AT | PR |
|---|---|---|---|---|---|
| 1 | VCC | – | Power supply | | |
| 2 | DOU | Output | UART data out | | |
| 3 | DIN / CONFIG | Input | UART data in | | / |
| 4 | DO8 | Output | Digital output | | |
| 5 | RESE | Input | Module reset (at least 200nS) | | |
| 6 | PWM0 / RSSI | Output | PWM output 0 / RX signal strength indicator | | |
| 7 | PWM | Output | PWM output | | |
| 8 | (reserved) | | Do not connect | D8 | |
| 9 | DTR / SLEEP RQ / DI8 | Input | Pin sleep control line or digital input 8 | | 6 |
| 10 | GND | – | Ground | | |
| 11 | AD4 / DIO4 | Either | Analog input 4 or digital I/O 4 | D4 | 0 |
| 12 | CTS / DIO7 | Either | Clear to send flow control or digital I/O 7 | D7 | |
| 13 | ON / SLEEP | Output | Module status indicator | | |
| 14 | VRE | Input | Voltage reference for AD inputs | | |
| 15 | Associate / AD5 / DIO5 | Either | Associated indicator, analog input 5 or digital I/O 5 | D5 | |
| 16 | RTS / AD6 / DIO6 | Either | RTS flow control, analog input 6 or digital I/O 6 | D6 | 5 |
| 17 | AD3 / DIO3 | Either | Analog input 3 or digital I/O 3 | D3 | 1 |
| 18 | AD2 / DIO2 | Either | Analog input 2 or digital I/O 2 | D2 | 2 |
| 19 | AD1 / DIO1 | Either | Analog input 1 or digital I/O 1 | D1 | 3 |
| 20 | AD0 / DIO0 | Either | Analog input 0 or digital I/O 0 | D0 | 4 |

Table 3.2.1 : Xbee Pin Configuration

## Xbee Pin Configuration and Electrical Specification

1. All Input/Output Lines : Eight of the I/O lines are fitted with pull-up resistors which can be enabled or disabled with the **PR parameter**. This is an 8-bit mask, where a 0 disables the pull-up and a 1 enables it. Table 3.3.2.1 gives the bit positions in the column labelled PR. The default value is 0xff.

   I/O pins are configured using the **Dn parameter**. The command is followed by the pin code from the AT column of Table 3.2.1, and the command assigns a mode code which is 0 to disable the line, 1 to enable the line's special function (see next para), 2 for analog input, 3 for digital input, 4 for digital output low and 5 for digital output high. I/O lines with special functions have those functions enabled by setting their **Dn parameter** to 1. These lines are line 7 (CTS), 6 (RTS) and 5 (Associated indicator). For lines 7 and 5, this is the default state. For line 6, the default state is disabled. Changes

made via the Dn parameter become effective the next time an AC command is issued.

2. Input/Output Data Packet : The **IS parameter** forces a read of all enabled inputs and outputs (analog or digital) with the data being returned through the UART. If no inputs are enabled, the command returns "ERROR". The IS command can take a parameter, with a value in the range 1 to 0xff and a default value of 1.

   An I/O data packet is also sent when the **IC parameter** has any bits set. This parameter enables change detection on DIO lines 7 to 0 when the corresponding bits are 1. Any change on any enabled line first causes transmission of any queued data, and then causes the DIO data to be transmitted. The default value for the IC parameter is 0x00. This packet will report only digital line states.

3. All Inputs : The IR parameter sets or reads the sample rate. When set to a value other than 0, the module will sample all enabled inputs at time intervals given by the parameter value. The time unit is mS, and the maximum value is 0xffff. The default is 0. The IT parameter sets or reads how many samples (max) will be transmitted per packet. Its value is in the range 1 to 0xff, with 1 as the default. When sleep modes are enabled and a sample rate (IR) is set, the module will remain awake until IT samples have been collected.

4. Digital Inputs : Digital inputs will recognize a voltage lower than $0.35 * \text{VCC}$ as low, or greater than $0.7 * \text{VCC}$ as high. The input leakage current for any input voltage between 0 and VCC is typically 0.025µA, max 1µA. The input leakage current spec is the same, whether or not the input is configured for high impedance. To configure I/O line n as a digital input, use the **Dn parameter** with value 3.

5. Digital Outputs : When a digital output is in the low state, it can sink up to 2mA with the voltage level being no higher than 0.5V. In the high state, it can source 2mA with the voltage being no lower than $\text{VCC} - 0.5$. To configure I/O line n as a digital output, use the Dn parameter with value 4 for low voltage, and 5 for high voltage. Line 8 cannot be used as a digital output in firmware versions earlier than 1xEx.

The IO parameter sets the output levels for lines configured as digital outputs. It's an 8-bit mask. The mask value overrides the Dn parameter, which remains as the default. The IO parameter has a curious side effect: if you give an IO command without a parameter value, it seems to behave as though you passed a value of zero, setting all digital outputs to their low state. Once in this state you can't restore an output whose Dn parameter was set to high just by re-issuing its Dn command. You have to set it low and then set it high again.

6. Analog Inputs and VREF : The voltage on any analog input pin must be at least VSSAD − 0.3 and at most VDDAD + 0.3. That's the electrical limit, not the range for the conversion to be valid. Any analog input must be filtered, with a capacitor in the range 0.01µF to 0.1µF connected between the input pin and VREF. Analog input pins must be driven from a source whose impedance has a real component of at most 10kΩ. An analog input pin whose voltage is VREF will be read as 0x3FF, as will any higher voltages.

   To configure I/O line n as an analog input, set the Dn parameter to 2. Only lines 0 to 5 can be used in this way.

   The voltage supplied to the VREF pin must be at least 2.08V and at most VDDAD. The module will draw 200µA from the source when A-D is enabled, or at most 0.02µA when A-D is disabled or the module is in sleep mode.

7. Analog Outputs : The PWM period is 64µS, and there are 1023 (0x3ff) steps within it. To put that another way, the PWM outputs deliver a square wave at a frequency of about 15.6KHz with a duty cycle that can be varied in 1024 steps between 0% and 100%. The Pp parameter sets the function of the PWMp pin. A value of 0 disables it, 1 sets it for RSSI and 2 sets it as a PWM output. The default is 1 for P0 and 0 for P1. In RSSI mode, the PWM output indicates received signal strength.

   The Mp parameter is the PWM output level for the PWMp pin. The value range is 0 to 0x03ff and the default is 0.

   The RP parameter gives the pulse duration for either PWM pin, when in RSSI mode. The duty cycle is updated with each received packet and is shut off when the timer expires. Timer values are in the range 0 to 0xff and the

time unit is 100mS. The default value is 0x28. See the product manual for further details.

8. Input/Output Line Passing : I/O line passing creates “virtual wires” between two Xbees, so that selected inputs on one (the source) are reflected as output states on the other (the destination).
9. Serial Lines : The serial lines operate at logic levels, with the idle state being high voltage. Each data byte is transmitted with one start bit (low voltage), eight data bits (low voltage for 0, high for 1, lsb first), an optional parity bit (but none by default) and one stop bit (high voltage). The baud rate is controlled by the BD parameter, which defaults to 3 (9600 baud).
10. Pin 5- RESET: This pin has a 50kΩ pull-up resistor within the module.

### 3.2.2 L293D

L293D is a popular motor driving IC. It is a 16 pin IC. The IC has 8 pins on both the sides. It has 2 enable pins, 1 $V_{SS}$ pin, 1 $V_S$ pin, 4 ground pins, 4 input pins and 4 output pins.

The descriptions of the pins are as follows:

1. Enable – the enable pins, when are given true**,** (i.e. 1) then they enable the respective part of the IC. The enable 1 chip enables the Left part of the IC for inputs and outputs, and so does the Enable 2 does to the right part of the IC.
2. VSS – this pin is to be given an input of 5 volts. This is used to power up the chip for its operations.
3. VS – This pin is given the voltage that we have to supply to the motors. This voltage comes out through the output pins. Due to the gates used in the IC, the output is usually 1.8 to 2 volts less than the Vs.
4. Input – the input pin decides whether output has to be given to the respective output pin or not. When the Input is true, then output is also 1 in the respective output pin. When input in the Input pin is 0, and then output in the respective output pin is also 0.
5. Output – the output pin is connected to the terminals of the motor. The input pins, as stated above, control its output.
6. GND **–** these pins are the ground pins, or, in other words, Zero.

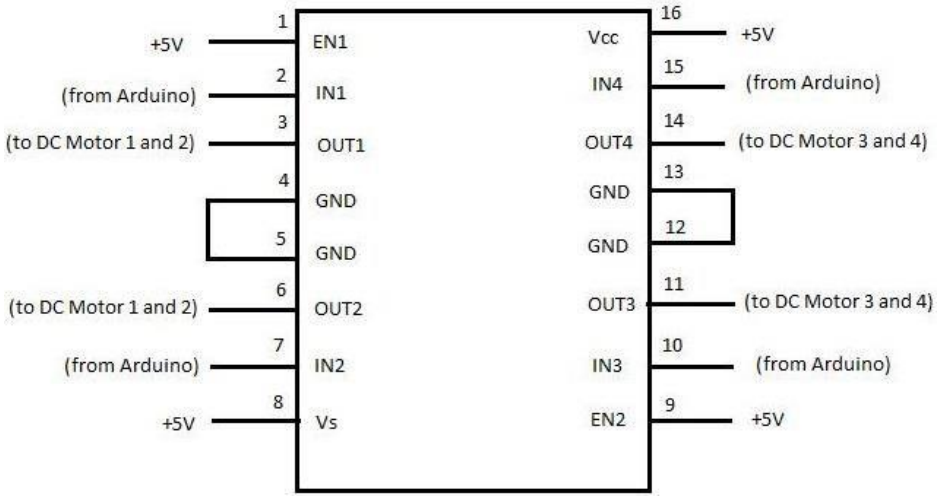



Figure 3.2.2 L293D pin configuration

### 3.2.3 ARDUINO UNO

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller. One can simply connect it with a usb cable or a ac-dc converter or a battery to start it.

| | |
|---|---|
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 14 of which 6 pins are PWM output |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |

| | |
|---|---|
| Flash Memory | 32 KB(ATmega328),0.5 KB used by boot-loader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Clock Speed | 16 MHz |

Table 3.2.3 Arduino Uno Specification

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

a) VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or,if supplying voltage via the power jack, access it through this pin.

b) 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

c) 3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

d) GND: Ground pins.

e) IOREF: This pin on the Arduino board provides the voltage reference

with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

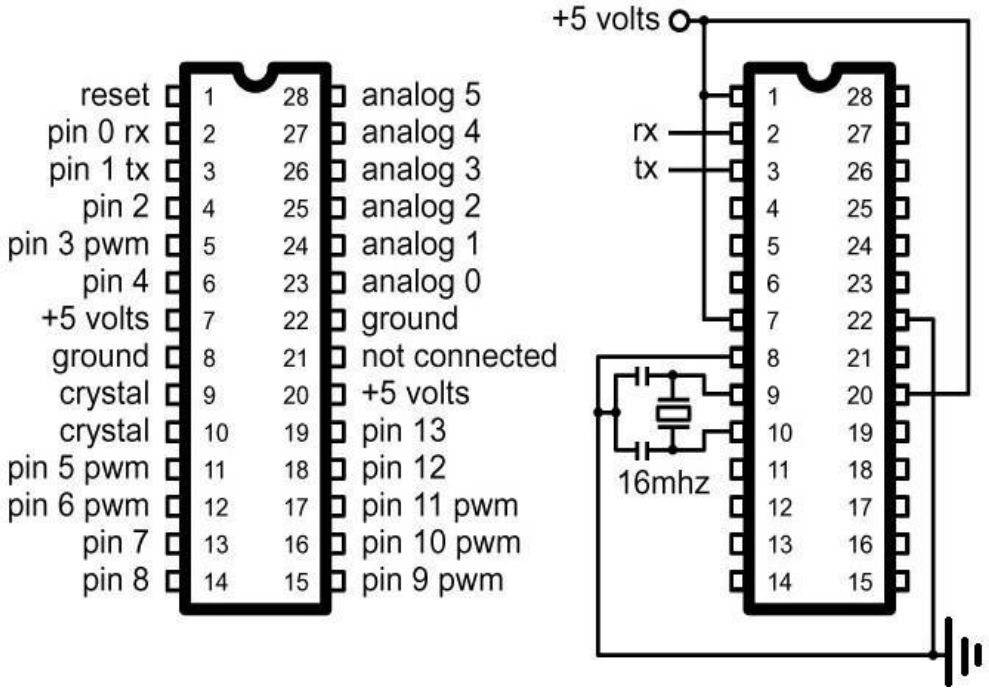



Figure 3.2.3 ATmega328 pin configuration

The ATmega328 has 32 KB (with 0.5 KB used for the boot-loader) memory. It also has 2 KB of SRAM and 1 KB of EEPROM. We programmed the Arduino using the Arduino software.

### 3.2.4 DC GEARED MOTOR

Geared DC motors can be defined as an extension of DC motor which already had its Insight details demystified here. A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. This Insight will explore all

the minor and major details that make the gear head and hence the working of geared DC motor.

**External Structure**

At the first sight, the external structure of a DC geared motor looks as a straight expansion over the simple DC ones. The lateral view of the motor shows the outer protrudes of the gear head. A nut is placed near the shaft which helps in mounting the motor to the other parts of the assembly. Also, an internally threaded hole is there on the shaft to allow attachments or extensions such as wheel to be attached to the motor.

### 3.2.5 7805 REGULATOR IC

It is used to convert the input varying supply (usually 9-18 volts) to a stabilized 5 volts supply, which is used to drive the circuitry. These are rated at a maximum current of 1A but they require heat-sinks when current exceeds 0.5A. 78XX series have become a popular choice due to their simplicity and convenience.

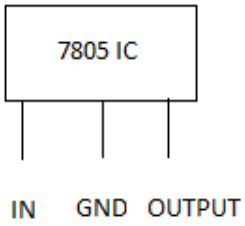



Figure 3.2.5 7805 pin diagram

### 3.2.6 TCRT SENSOR

TCRT is obstacle detecting sensor which works on the basis of light sources. It has a compact construction where the emitting-light source and the detector are arranged in the same direction to sense the presence of an object by using the reflective IR beam from the object. The operating wavelength is 950 mm. The detector consists of a phototransistor.

**Physical Description:**

- Snap-in construction for PCB mounting.
- Package height: 7 mm
- Plastic polycarbonate housing construction which prevents crosstalk.

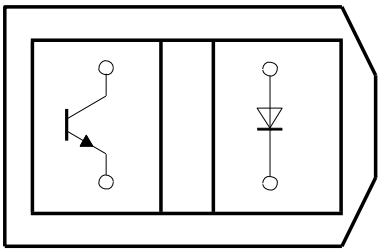


Figure 3.2.6 Top view of TCRT sensor

**Application:**

- Position sensor for shaft encoder.
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR.
- Used as edge detection sensor in our project.

### 3.2.7 SONAR (HCSR04)

It is an Ultrasonic ranging sensor which has the ranging accuracy up to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit.

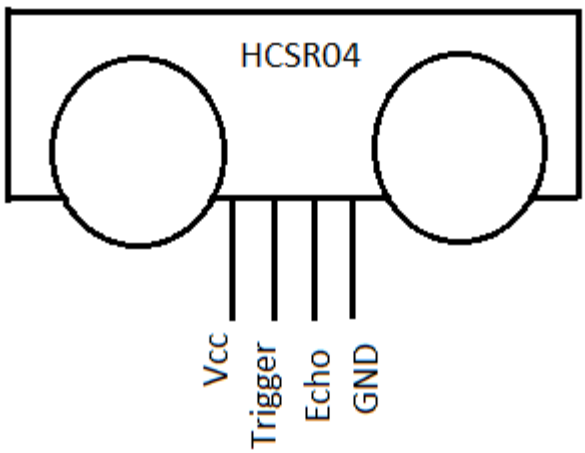



Fig 3.2.7 Sonar sensor

Using IO trigger for at least the module automatically sends eight 40 kHz and detect whether there is a pulse signal back. If the signal is received back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Its pin consists of

- 5V Supply Pin
- Trigger Pulse Input Pin
- Echo Pulse Output Pin
- 0V Ground Pin

### 3.2.8 VACUUM CLEANER

We were provided with small dc vacuum, which we modified as per our requirements. The vacuum in our project consists of suction motor, fan and a suction pipe. We made suction pipe with the help of can-bottle and modified it accordingly to suck the dirt and dust from the cleaning area. By providing small voltage (up to 5v), the vacuum was successfully able to suck the dust to clean the surface.

The vacuum cleaner works on the basis that the fan inside the vacuum causes the air pressure inside the vacuum cleaner to drop, hence air including dust particles moves to the generated suction, a partial vacuum through the intake port to the exhaust port.

### 3.2.9 IP CAMERA

An Internet protocol camera, or IP camera, is a type of digital video camera commonly employed for surveillance, and which unlike analog closed circuit television (CCTV) cameras can send and receive data via a computer network and the Internet. Although most cameras that do this are webcams, the term "IP camera" or "netcam" is usually applied only to those used for surveillance. The first centralized IP camera was Axis Neteye 200, released in 1996 by Axis Communications.

Benefits of IP camera over analog technology include:

- Remote administration from any location.
- Digital zoom.
- The ability to easily send images and video anywhere with an Internet connection.
- Progressive scanning, which enables better quality images extracted from the video, especially for moving targets.
- Adjustable frame rates and resolution to meet specific needs.
- Two-way communication.
- The ability to send alerts if suspicious activity is detected.
- Lower cabling requirements.
- Support for intelligent video

## 3.3 SOFTWARE DESCRIPTION

### 3.3.1 MICROSOFT VISUAL BASIC

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop console and graphical user interface applications. Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger.

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++, VB.NET, C# and F#.

Visual Studio features background compilation. As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code.

We created a simple application using Visual Studio for sending data through our laptop to the RF transmitter via serial communication

### 3.3.2 ARDUINO PROGRAMMING

We programmed the Arduino board for control of vacuum cleaner as well as geared DC motors and for the action of the transmitted data on the car. The TCRT sensor and Sonar sensor were also programmed using arduino programming for the desired operation.

# CHAPTER 4
# WORKING PRINCIPLE

Vacuum cleaner attached to a robotic vehicle is the basics of RVC. Robotic vacuum cleaner is based on automatic and manual mode of operation. The sensor value and incoming radio frequency signal are processed by the microcontroller for desired output operation.

## Vacuum Operation

Vacuum cleaner is attached to the lower part of robot adjusting some ground clearance. Vacuum consists of electric motor, fan and a dust collector. Suction operation is carried out by the electric motor and fan. When the motor is driven, the pressure difference between front and back portion of the fan is distinguishable. As a result suction, partial vacuum, is created inside the vacuum cleaner and the ambient air is pushed along with the dust particle to the intake port, ultimately gathering to dust collector.

## Xbee Mode of Operation

The robot with vacuum cleaner is made to move on the subjected area by the manual inspection viewing the camera feedback. Wireless manual feedback is provided to the robot by using radio frequency modules (Xbee Modules pro Series 2), attached on receiver and transmitter sides. The vehicular control signal depends on the operation of motors attached on the robot. And the vacuum mode operation is dependent on the operation of the next motor controlling the fan movement, attached on the lower part of robot. These motors are controlled by using keyboard buttons of the computer.

## Xbee Configuration

Xbee configuration is carried by using special software called X-CTU. Xbee is flashed with latest firmware versions available in X-CTU software. One of the Xbee is configured in routing mode whereas next one is configured in coordinator/end device mode using same Personal Authentication Number (PAN). After the configuration, xbees connected in either side on microcontroller (arduino) are ready for UART communication and the data from keyboard can be serially received on receiving side.

## Camera Configuration

The use of IP camera enables remotely access of the subjected area. IP camera is configured using IP address of the camera connecting to a DSL router. After the configuration, the live streaming of video camera can be obtained on screen of the computer remotely. Use of camera enables the inspection of subjected surface area manually. And the mode of operation can be triggered to manual mode for residual cleaning to obtain better experience of cleaning process by RVC.

## Automatic Mode of Operation

The automatic mode operation is possible only if the robot has the outside world information access. The sensors act as a window providing the robot access to outside world. Automatic mode includes edge detection using TCRT sensors and obstacle detection using sonar sensor (HC-SR04). IR diode and TSOP are bundled together to form TCRT sensor which is basically used for color detection of the surfaces. TCRT is attached to the lower part of vehicle facing towards the ground surface with separation of 3cm from the surface. We use TCRT sensor values for the edge detection and feedback is provided to the Arduino input port which in-turn is programmed in such a way to prevent vehicle from falling off edge, thus maintaining edge detection automatically. HC-SR04 works on the principle of emission and reflection of ultrasound, calculates the distance of distant obstacle computing the time delay and sound velocity towards a certain direction. The echo and trigger pins of HC-SR04 are connected to input pins of Arduino and the obstacle location is identified through the program. The lines of code are written to avoid the obstacle by driving the motors in suitable manner and the reposition of vehicle is carried afterward.

# CIRCUIT DIAGRAM

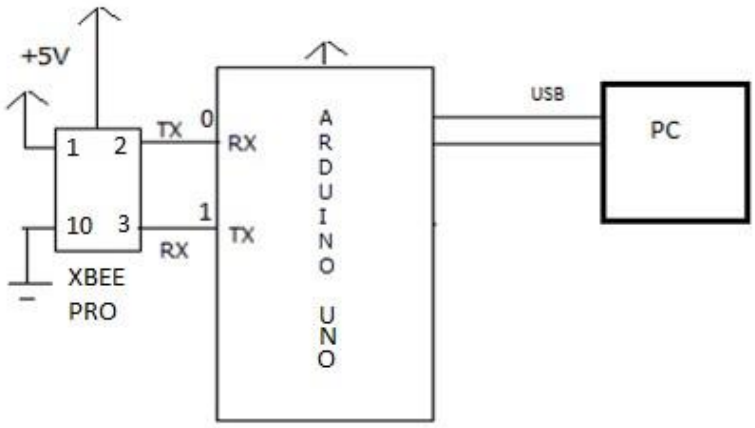



Fig : 9.1 Transmitter Circuit

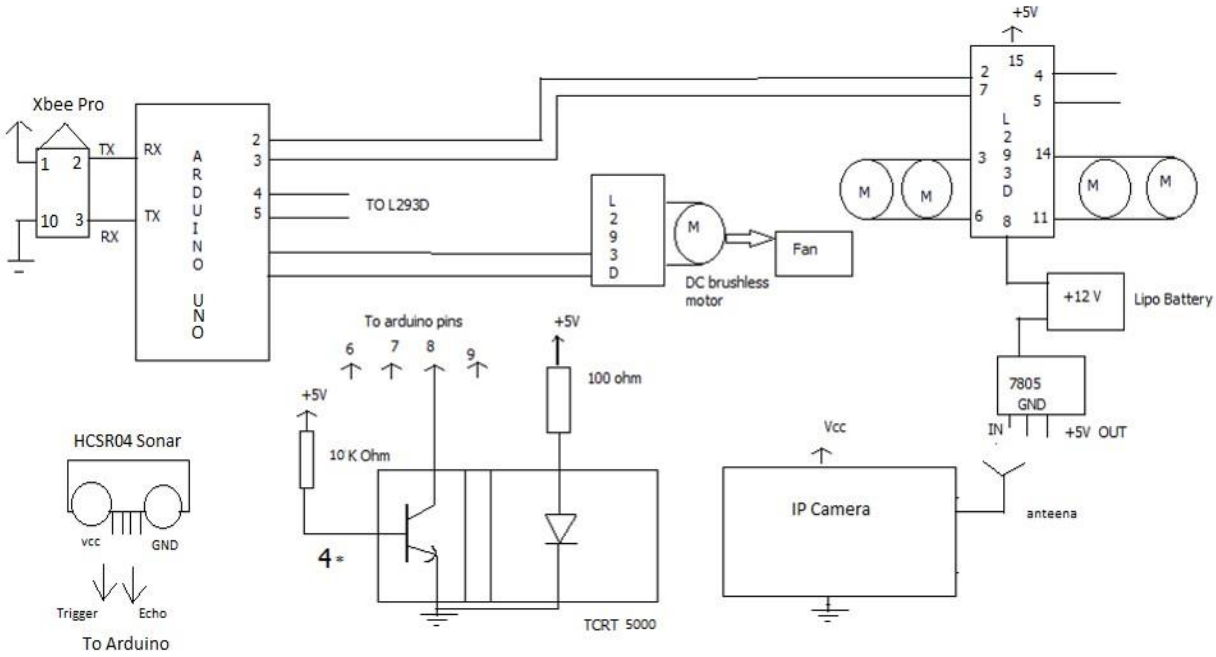



Fig : 9.2 Receiver Circuit

# CHAPTER 5

# APPLICATION OF ROBOTIC VACUUM CLEANER

- **For household purposes:**
  It can be used for reducing the human contact with dust.
- **For agriculture purposes:**
  The robot can be useful in the field of agriculture. It can be used for collecting grains from soils.
- Robotic Vacuum Cleaner can be used in cleaning purposes in commercial areas like restaurant, hospitals, shopping malls, highways, etc.
- It can be integrated with a powerful drier to dry up a wet surface.

# CHAPTER 6

# CONCLUSION

Our Project entitled “Robotic Vacuum Cleaner” was successfully completed as per our expectations and specifications. The operation of surface cleaning, obstacle avoidance and edge detection and avoidance was also successfully conducted. This project was started with the objective to embed vacuum cleaner in a robotic car and use it in various places for cleaning as well as survellience using video feedback, assisting humans by avoiding direct human contact with dust.

Our project in wireless communication of RF module gave us great knowledge about the serial communication, RF transmission and reception, motor driving mechanism and sensor based programming and operation. We have completed a prototype of a cleaning robot despite of various complexities and also leaving various options behind for further enhancements.

# CHAPTER 7

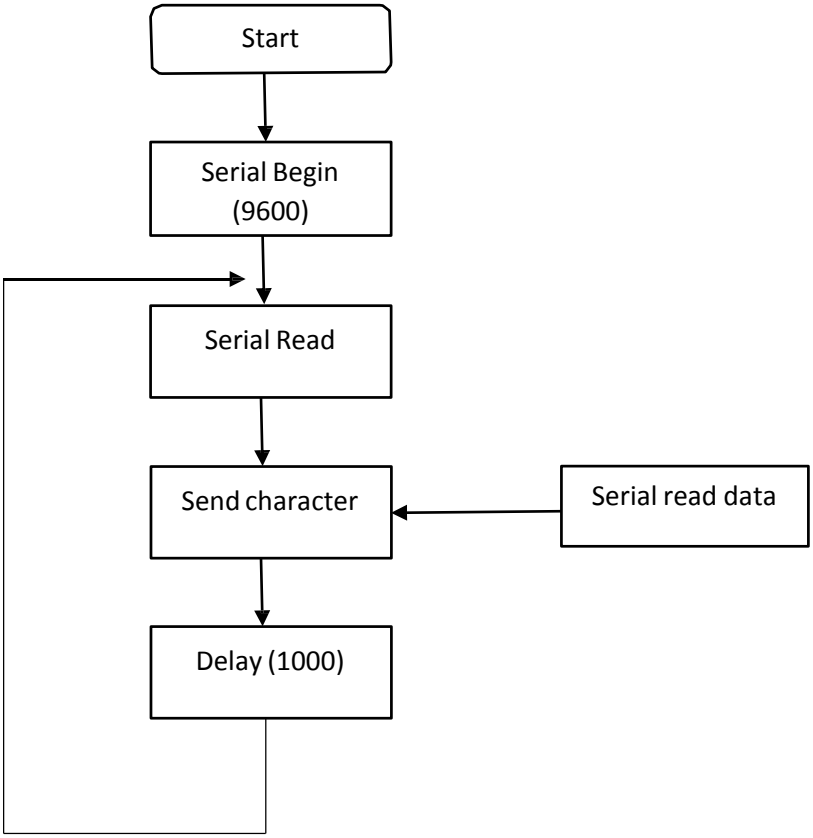
# RECOMMENDATION AND FUTURE ENHANCEMENT

1. We can use high powered RF module for long distance wireless communication
2. We can use high powered torque motors in order to provide sufficient torque to drive motors under heavy loaded condition
3. Robotic arm can be added to the vehicle in order to pick up waste which are bigger in size than the capacity of the vacuum cleaner
4. We can develop a scheduler to program the robot to automatically clean at scheduled times.
5. A concept of home base can be developed so that the robot can be programmed to return and dock to its charging station after completing its task
6. Our vehicle can be enhanced to a voice controlled locomotion using advanced speech recognition software.

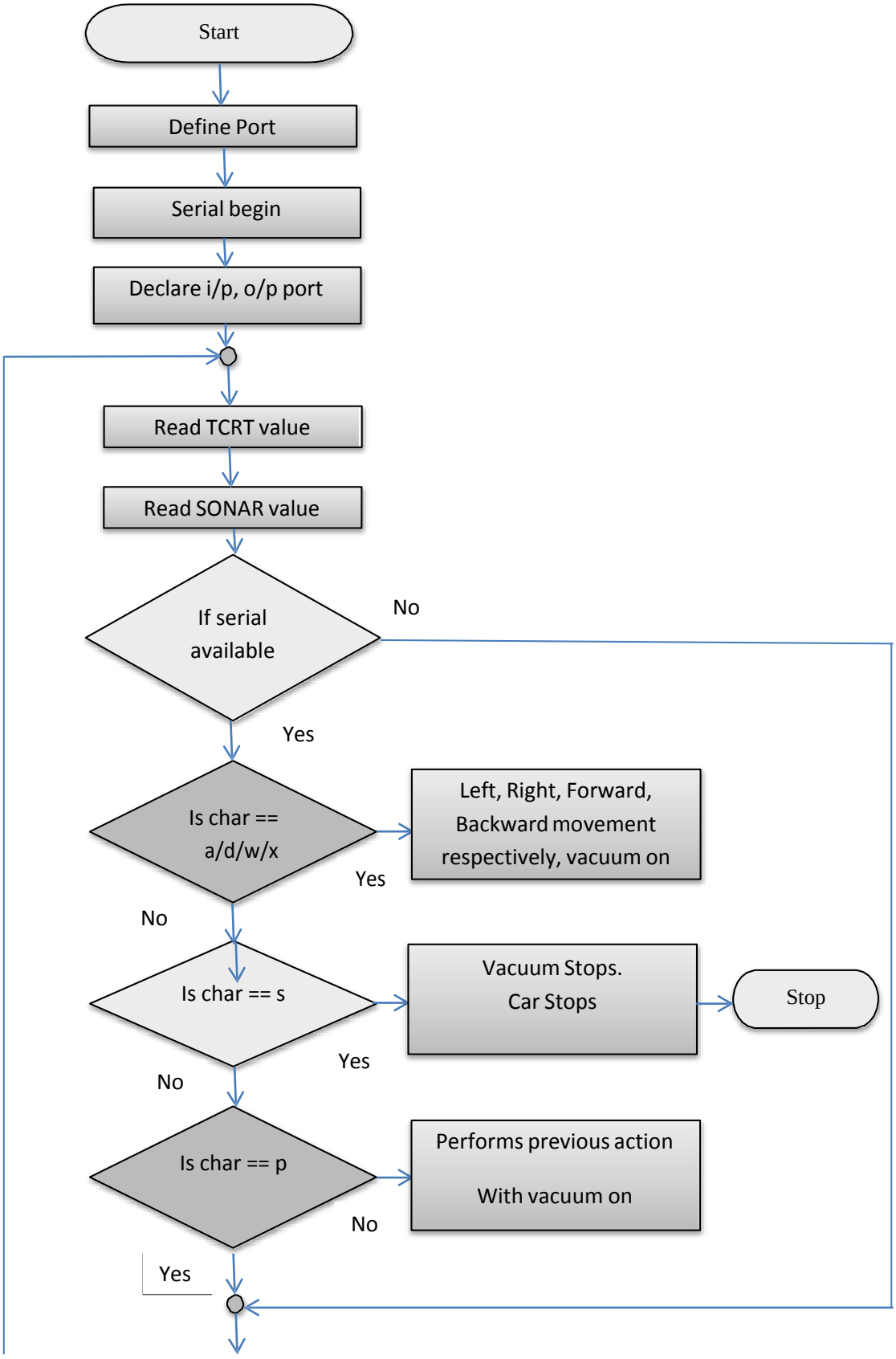
# CHAPTER 8
# REFERENCES

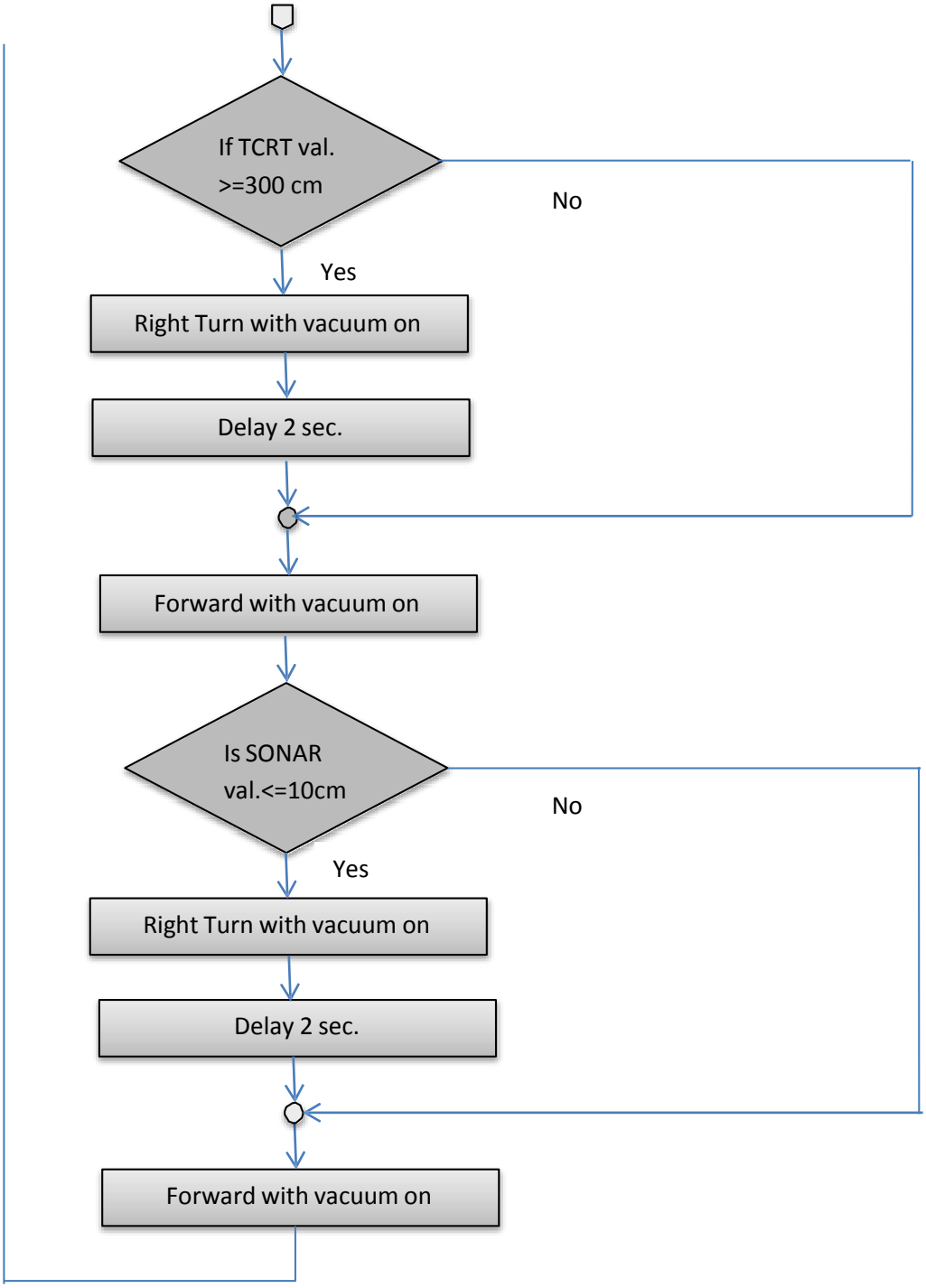

1. http://www.jbprojects.net/projects/wifirobot/http://www.codeproject.com/
2. http://www.csse.monash.edu.au/hons/projects/1999/Tim.Bruton
3. http://www.arduino.cc
4. http://extremeelectronics.co.in/
5. http://www.instructables.com/id/Xbee-quick-setup-guide-Arduino/
6. http://www.electroschematics.com/8902/hc-sr04-datasheet/
7. http://www.datasheetcatalog.com/datasheets_pdf/T/C/R/T/TCRT5000.shtml
8. http://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/
9. http://arduinobasics.blogspot.com/2012/11/arduinobasics-hc-sr04-ultrasonic-sensor.html
10. http://robotic-controls.com/learn/wireless-communication/xbee-configuration
11. http://blog.huntgang.com/2014/06/17/arduino-tcrt5000-build-ir-sensor/
12. http://www.instructables.com/id/Processing-Controls-RC-Car-with-XBee-modules/?ALLSTEPS
13. Michael Margolis(2001), *Arduino Cookbook*, Second Edition, Sebastopol, O'Reilly Media
14. David R Shircliff, *Build a Remote Controlled Robot.* [Online], McGraw-Hill. Available from: http://www.robotbooks.net/build-a-remote-controlled-robot-david-r-shircliff-11007/
15. Sedra and Smith (1998), *Microelectronic Circuit*, Fourth Edition, London, Oxford University Press

# FLOWCHART




Transmitter Flowchart


Start
Define Port
Serial begin
Declare i/p, o/p port
Read TCRT value
Read SONAR value
If serial available
No
Yes
Is char == a/d/w/x
Left, Right, Forward, Backward movement respectively, vacuum on
Yes
No
Is char == s
Vacuum Stops. Car Stops
Stop
Yes
No
Is char == p
Performs previous action With vacuum on
No
Yes




Receiver Flow Chart

# APPENDIX

# PCB DIAGRAM

1. L293D Motor Driver

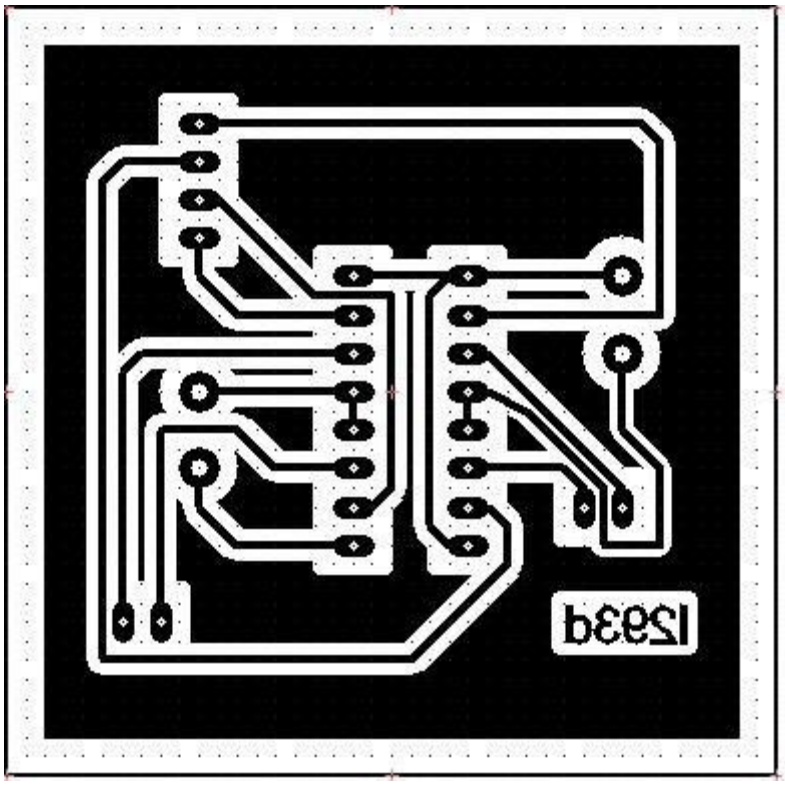



Fig : L293D

2. Sonar Module HCSR04

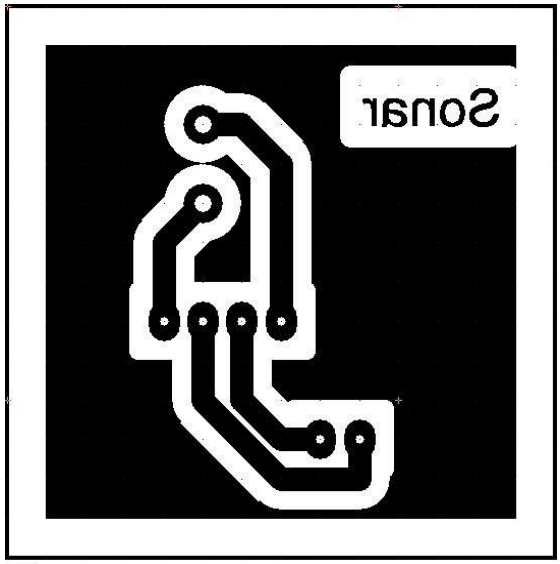



Fig : Sonar

3. TCRT Module

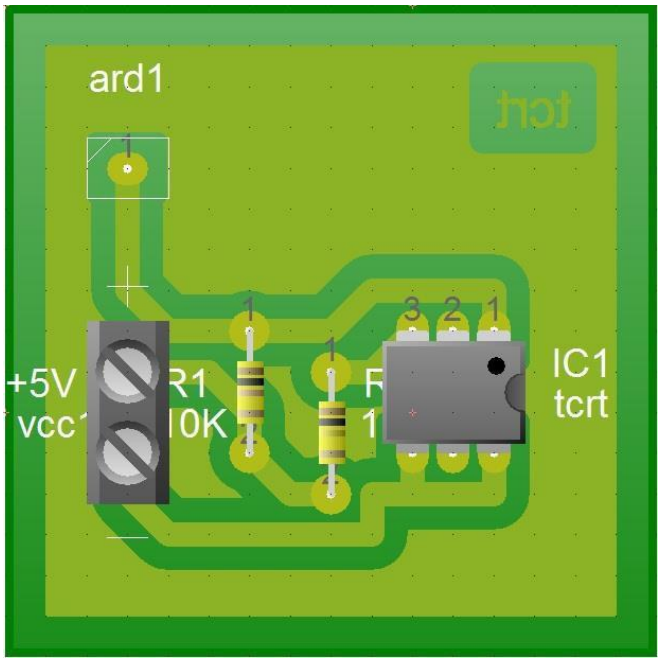



Fig : TCRT

3. Voltage Regulator

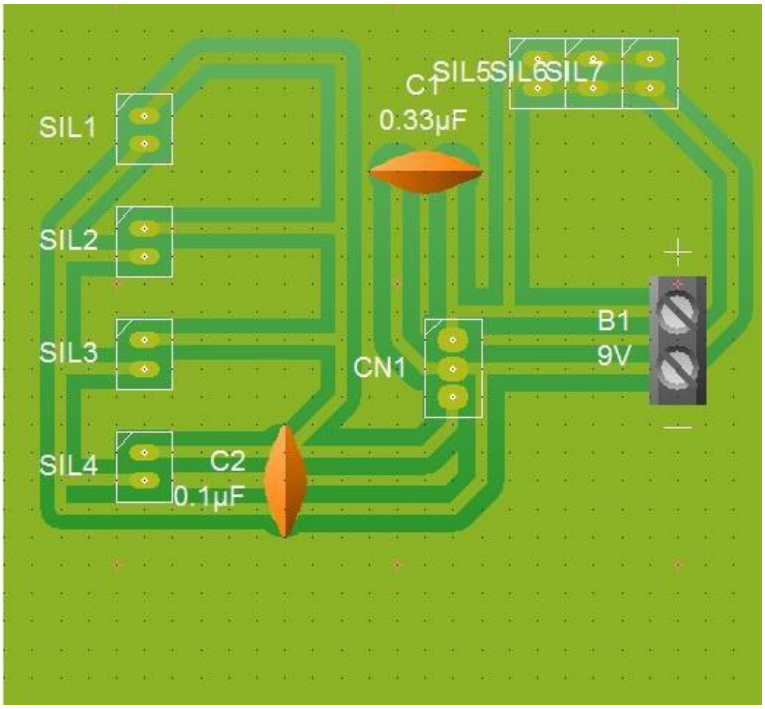



Fig : Voltage Regulator

# Ultrasonic Ranging Module HC - SR04

## Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

## Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

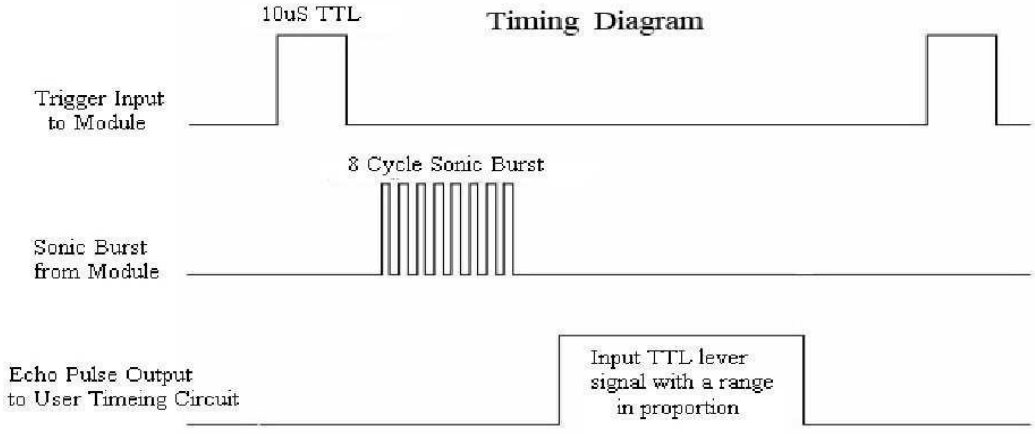
## Electric Parameter

| | |
|---|---|
| **Working Voltage** | **DC 5 V** |
| **Working Current** | **15mA** |
| **Working Frequency** | **40Hz** |
| **Max Range** | **4m** |
| **Min Range** | **2cm** |
| **MeasuringAngle** | **15 degree** |
| **Trigger Input Signal** | **10uS TTL pulse** |
| **Echo Output Signal** | **Input TTL lever signal and the range in proportion** |
| **Dimension** | **45*20*15mm** |


**Vcc Trig Echo GND**

## Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



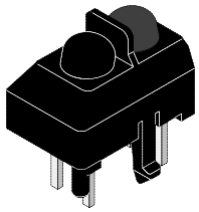
## Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

# Reflective Optical Sensor with Transistor Output

## Description

The TCRT5000(L) has a compact construction where the emitting-light source and the detector are arranged in the same direction to sense the presence of an object by using the reflective IR beam from the object. The operating wavelength is 950 mm. The detector consists of a phototransistor.

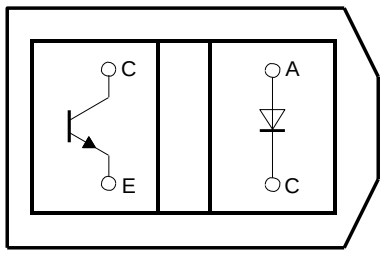


94 9442

## Applications

- Position sensor for shaft encoder
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR
- General purpose – wherever the space is limited

15116




Top view

## Features

- Snap-in construction for PCB mounting
- Package height: 7 mm
- Plastic polycarbonate housing construction which prevents crosstalk
- L = long leads
- **C**urrent **T**ransfer **R**atio (CTR) of typical 10%

## Order Instruction

| Ordering Code | Sensing Distance | Remarks |
|---|---|---|
| TCRT5000 | 12 mm | Leads (3.5 mm) |
| TCRT5000(L) | 12 mm | Long leads (15 mm) |

## Absolute Maximum Ratings

### Input (Emitter)

| Parameter | Test Conditions | Symbol | Value | Unit |
|---|---|---|---|---|
| Reverse voltage | | $V_R$ | 5 | V |
| Forward current | | $I_F$ | 60 | mA |
| Forward surge current | $t_p$ δ 10 µA | $I_{FSM}$ | 3 | A |
| Power dissipation | $T_{amb}$ δ 25 C | $P_V$ | 100 | mW |
| Junction temperature | | $T_j$ | 100 | C |

### Output (Detector)

| Parameter | Test Conditions | Symbol | Value | Unit |
|---|---|---|---|---|
| Collector emitter voltage | | $V_{CEO}$ | 70 | V |
| Emitter collector voltage | | $V_{ECO}$ | 5 | V |
| Collector current | | $I_C$ | 100 | mA |
| Power dissipation | $T_{amb}$ δ 55 C | $P_V$ | 100 | mW |
| Junction temperature | | $T_j$ | 100 | C |

### Sensor

| Parameter | Test Conditions | Symbol | Value | Unit |
|---|---|---|---|---|
| Total power dissipation | $T_{amb}$ δ 25 C | $P_{tot}$ | 200 | mW |
| Operation temperature range | | $T_{amb}$ | –25 to +85 | C |
| Storage temperature range | | $T_{stg}$ | –25 to +100 | C |
| Soldering temperature | 2 mm from case, t δ 10 s | $T_{sd}$ | 260 | C |

## Electrical Characteristics ($T_{amb}$ = 25°C)

### Input (Emitter)

| Parameter | Test Conditions | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Forward voltage | $I_F$ = 60 mA | $V_F$ | | 1.25 | 1.5 | V |
| Junction capacitance | $V_R$ = 0 V, f = 1 MHz | $C_j$ | | 50 | | pF |

### Output (Detector)

| Parameter | Test Conditions | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Collector emitter voltage | $I_C$ = 1 mA | $V_{CEO}$ | 70 | | | V |
| Emitter collector voltage | $I_E$ = 100 µA | $V_{ECO}$ | 7 | | | V |
| Collector dark current | $V_{CE}$ = 20 V, $I_F$ = 0, E = 0 | $I_{CEO}$ | | 10 | 200 | nA |

### Sensor

| Parameter | Test Conditions | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Collector current | $V_{CE}$ = 5 V, $I_F$ = 10 mA, D = 12 mm | $I_C$ [1,2)] | 0.5 | 1 | 2.1 | mA |
| Collector emitter saturation voltage | $I_F$ = 10 mA, $I_C$ = 0.1 mA, D = 12 mm | $V_{CEsat}$ [1,2)] | | | 0.4 | V |
| [1)] See test circuit | | | | | | |
| [2)] Test surface: Mirror (Mfr. Spindler a. Hoyer, Part No 340005) | | | | | | |

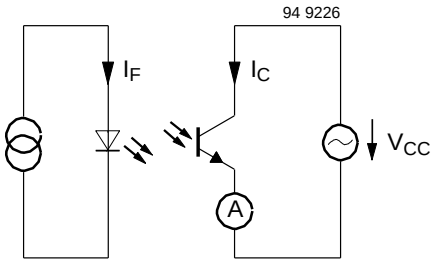



Figure 1. Test circuit

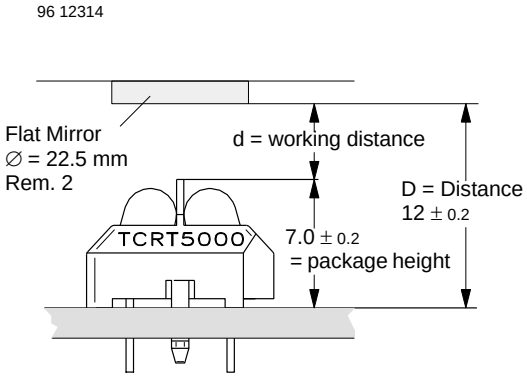



Figure 2. Test circuit

## Typical Characteristics ($T_{amb}$ = 25 C, unless otherwise specified)

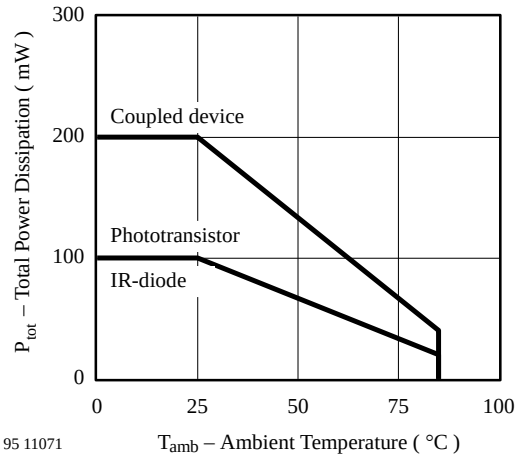



Figure 3. Total Power Dissipation vs. Ambient Temperature

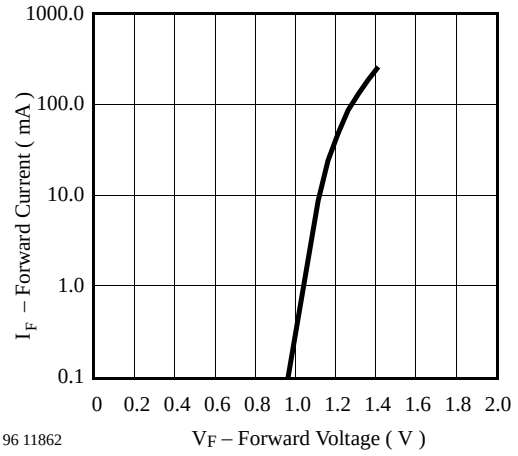



Figure 4. Forward Current vs. Forward Voltage

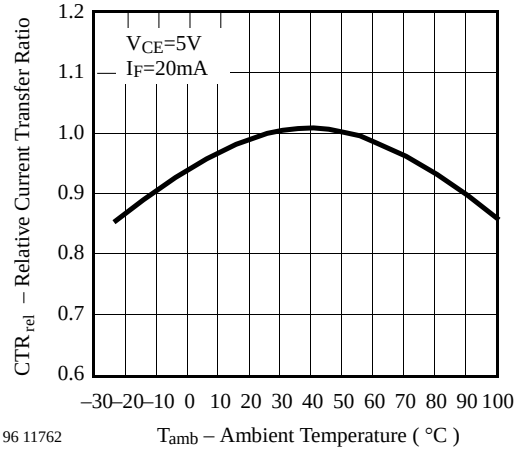



Figure 5. Rel. Current Transfer Ratio vs. Ambient Temp.

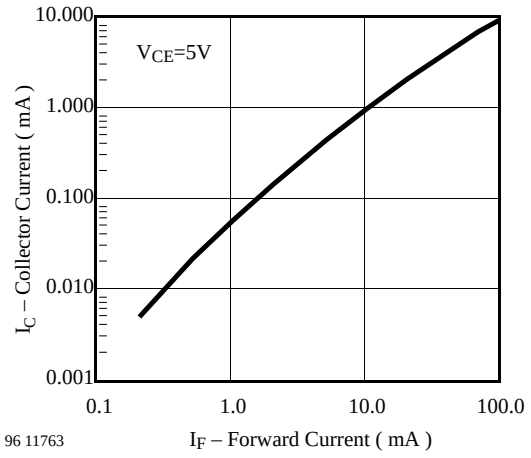



Figure 6. Collector Current vs. Forward Current

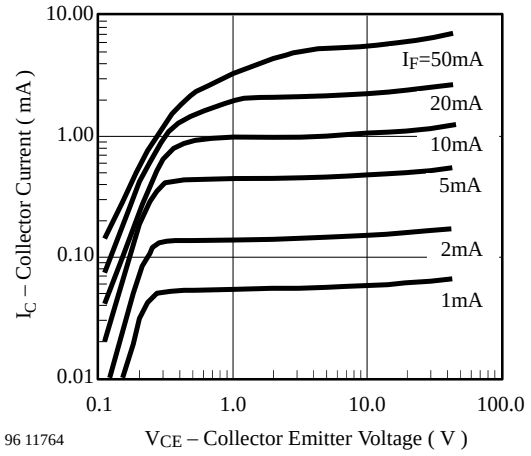



Figure 7. Collector Emitter Saturation Voltage vs. Collector Current

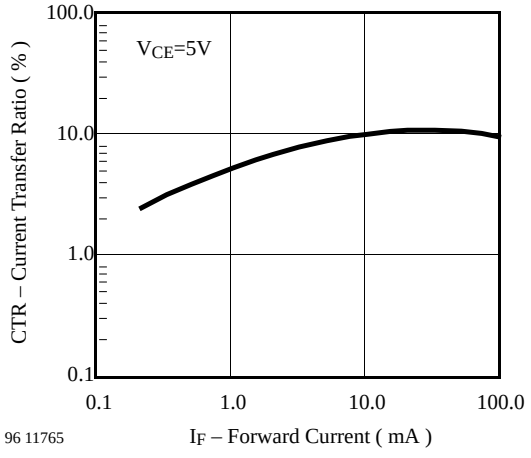



Figure 8. Current Transfer Ratio vs. Forward Current

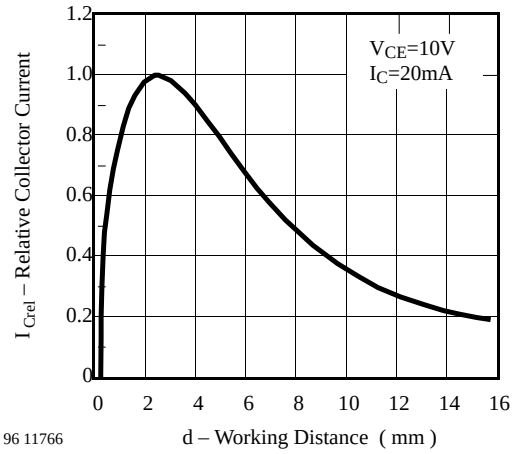



Figure 9. Relative Collector vs. Distance

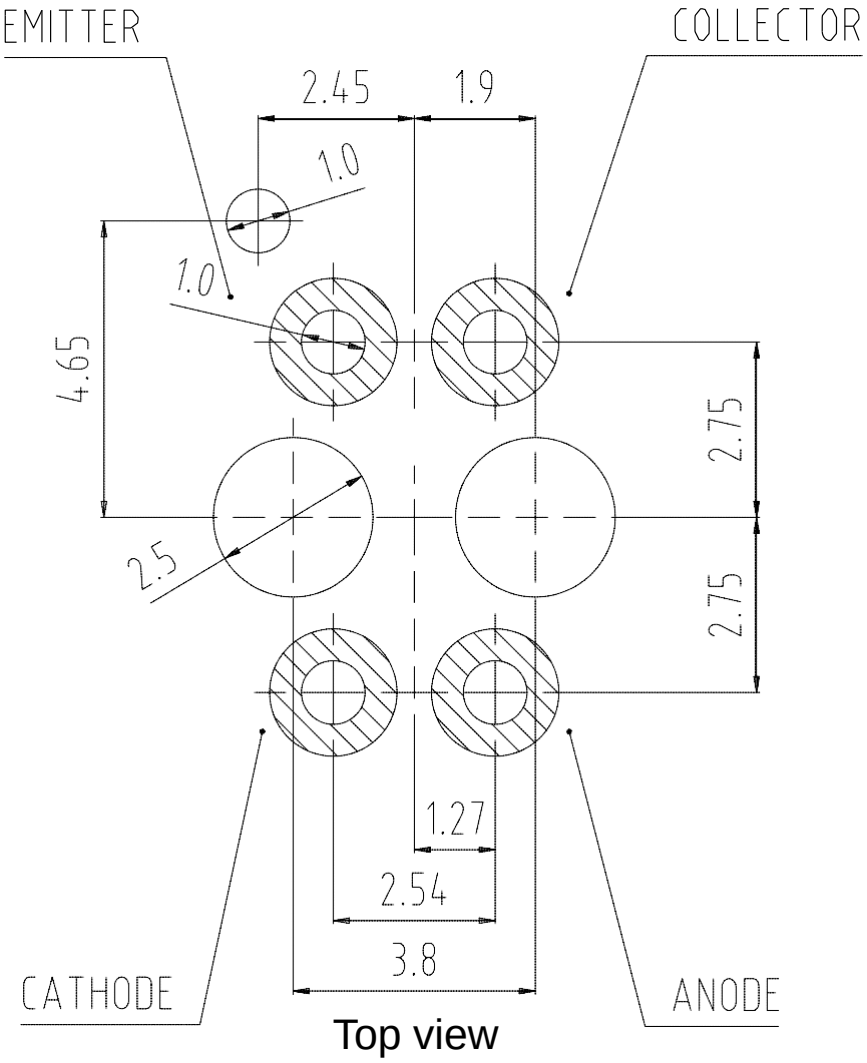



Figure 10. Footprint

# Operation

## 3.1 Version and Device Information

The **VR command** returns the firmware version of the RF module, as a value in the range 0 to 0xffff.

The **VL command** used to return a verbose description of versions, including the application build date, MAC, PHY and bootloader versions. Under firmware version 10C8, this command cannot be used as an API command. If you try it as a local command it returns error status, and as a remote command it gets no response at all. In fact, the 1xEx product manual says the command was removed from firmware version 1xC9 and all later versions.

The **HV command** returns the hardware version of the RF module, in the range 0 to 0xffff. On my ordinary XBees it returns 1742 (1744 on a few that I bought more recently), and on the Pro it returns 1842. A forum post by gworle of Digi says that plain series 1 XBees will return 17xx and Pros will return 18xx. For the series 2 modules, a plain one will return 19xx and a Pro will return 1axx.

The **DD command** appears to be new in firmware version 1xCx. According to X-CTU, it returns the "device type identifier value". On my XBees the value is 0x10000, for both the normal and Pro versions. The DD command is not mentioned in the 1xCx product manual, nor in the 1xEx manual.

The **CK command** was introduced in version 10e4 according to the firmware revision history. It isn't mentioned in the product manual for 10e6. It returns the checksum of the configuration in RAM, as an aid to determining whether the configuration has changed. The value is a single byte.

## 3.2 Module Mode

The module mode controls how the XBee responds to data received via its UART. There are two possible modes: transparent (the default) and API. In API mode there are two sub-modes, allowing data to be sent as straight binary or with certain characters escaped. The **AP command** selects the mode. A value of 0 (default) selects transparent mode, 1 enables API mode, and 2 enables API mode with escaped control characters.

### 3.2.1 Transparent Mode

Transparent mode is enabled by default. Any RF data received by the module will be transmitted out through the UART, and any data received from the UART will be transmitted over RF. The module has a limited buffer size for this data (100 bytes), so with high transfer rates flow control must be implemented.

The **RO command** affects the packetization timeout. Its value lies in the range 0 to 0xff, and it gives the number of character times to wait after the last character was received, before transmitting the received data as a packet. When the value is 0, all characters received are transmitted immediately. The default value is 3.

### 3.2.2 API Mode

In API mode, all data entering and leaving the module is contained in frames that cause operations or events within the module. Frames received through pin 3 (DI) are called Transmit Data Frames. Frames sent out through pin 2 (DO) are called Receive Data Frames.

From the command state, the **AP command** enables or disables API mode. A value of 0 disables it, 1 enables it, and 2 enables it with escaped control characters. The default value is 0. Operation with escaped control characters would be necessary if using XON/XOFF flow control.

## 3.3 Module State

At any given time, a module is in one of five states: idle, transmit, receive, command or sleep. From the idle state it can switch to any other state, and from any other state it can switch back to idle.

### 3.3.1 Idle State

From the idle state:

- Serial data received in the DI buffer will switch the state to transmit.
- Valid data received through the RF antenna will switch to receive.
- If the sleep mode condition is met, the module will switch to sleep.
- If a command mode sequence is issued, the module will switch to command.

### 3.3.2 Sleep State

Note that in my application so far I don't use sleep modes. The information given here is therefore sketchy compared to what the product manual contains.

Sleep is a state in which power consumption is low. For sleep mode to be entered, sleep must be enabled using the **SM parameter**, and either pin 9 (Sleep_RQ) must be asserted or the module must have been idle for the length of time specified by the **ST parameter** (Time Before Sleep). The parameter gives the time in milliseconds. It takes values in the range 1 to 0xffff, and the default value is 0x1388 (5 seconds).

Sleep is disabled by setting SM=0. There are four other SM values: SM=1 gives Pin Hibernate, SM=2 gives Pin Doze, and SM=4 or SM=5 give Cyclic Sleep. If SM=5, the module will wake up after the timer expires or on a pin transition.

Pin Hibernate is the sleep mode with the lowest power consumption. When it is enabled, sleep is controlled solely by the state of the SLEEP_RQ pin. The wake-up time is 13.2mS.

Pin Doze has higher power consumption than pin hibernate, but has a faster wake-up time of 2mS. It too is controlled solely by the state of the SLEEP_RQ pin.

Cyclic Sleep has the same current consumption as pin doze. Sleep is entered after a no-activity timeout given by the **ST parameter**, and the module wakes after a period given by the **SP parameter**. While asleep, the RF module wakes periodically to detect whether RF data is present. The SP parameter takes values in the range 0 to 0x6880, with the time unit being 10mS. The default setting for SP is 0. The maximum value of 0x68b0 gives 268 seconds (4m28s). SP values for the coordinator and the end devices should be equal.

For end devices that are configured for association but not currently associated, the **DP command** controls the sleep time, with the same unit and range as for SP but with a default of 0x3e8 (10 seconds).

When a module awakes from sleep, it will sample all active I/O lines and transmit them. If the **IR parameter** is set, it will collect extra samples until the limit set by the **IT parameter** is reached. This automatic sampling on wakeup may be suppressed by setting bit 1 of the **SO parameter** (firmware 10cx).

The 1xAx product manual says that in all sleep modes, the current consumption rises dramatically if the supply voltage is more than 3.0V. In the 1xCx product manual, however, there is no mention of this.

### 3.3.3 Command State

In the command state, incoming characters are interpreted as commands. There are two sub-states here: AT command state and API command state.

To enter the AT command state, send the three-character sequence "+++" with a guard interval before and after it. The guard interval is determined by the **GT parameter**: if it is set to 0x3e8 (the default), the guard interval is one second. The time unit is milliseconds, and the value range is 2 to 0x0ce4. The three + characters must also be sent within a time equal to the guard interval. The **CC parameter** specifies the ASCII code of the character to be recognized where '+' is written above; '+' is the default.

When in AT command state, commands take the form of AT followed by the command name (two letters, or a letter and a digit), followed by an optional space and then an optional hex parameter value. Each command is terminated by a carriage return character or a comma. If the hex parameter value is omitted, the current value will be displayed. After successful execution of a command, the module will display “OK”, or the result if the command was to display a parameter value. If an error occurred, it will instead display “ERROR”.

The **AC command** will explicitly apply changes to queued parameter values and then re-initialize the module. After this command is given, the module will be operating according to the parameter values.

Stored values will not be retained after a reset, unless the **WR command** is issued. The WR command copies parameter values to non-volatile memory. It takes a little time: don’t send any other command until the “OK” has been received.

To restore all parameters to their factory default values, use the **RE command**. This command does not copy the restored values to non-volatile memory, so use the WR command afterwards if that’s required.

To perform a software reset, use the **FR command**. It will respond immediately with OK, and then perform the reset about 100mS later. The FR command was introduced in firmware version 1x80, and is equivalent to powering off and then on again.

To exit from AT command state, either send the **CN command** or wait for a period controlled by the **CT parameter**. The CT parameter is a timeout value in the range 2 to 0x1770 (10 minutes), and the time unit is 100mS. The default value is 0x64 (10 seconds). Note: before firmware version 10e4 the maximum setting was 0xffff and the 10e6 product manual still has that figure. The change is noted in the firmware revision history.

## 3.4 Flow Control

Flow control is necessary on any serial line if the rate of transmission is such that the receiving buffer cannot always be guaranteed to be emptied at a rate which prevents it from overflowing. There are two flow control mechanisms: hardware flow control and software flow control.

- Hardware flow control is used between an XBee and its host.
- Software flow control is used between two hosts that are communicating via XBees. The XBees themselves do not take any notice of the software flow control characters.[1]

Whether you need to worry about flow control at all is entirely dependent on your application.

### 3.4.1 Hardware Flow Control

The hardware flow control mechanism (between host and XBee) uses the RTS and CTS serial lines. The RTS line is used by the host, to signal to the XBee that the host’s buffer is nearly full and the XBee should stop transmitting until the line is de-asserted. In the same way, the XBee uses the CTS line to pause transmissions from the host.

When either flow control line is activated, any character already being transmitted will still be sent, and in general more than one character may be received after the line activation. Software needs to be written in such a way as to allow for this. It’s the reason why the lines need to be activated *before* the respective buffers are full, instead of when they *are* full.

### 3.4.2 Software Flow Control

With software flow control (host to host), two ASCII characters are given special meanings. The character 0x13 (Control-S or DC3 or XOFF) when sent across the serial link tells the other side to stop sending. The character 0x11 (Control-Q or DC1 or XON) tells the other side to resume sending. When software flow control is in use, it follows that these characters cannot be sent as part of the data stream.

As with hardware flow control, the programmer cannot assume that characters will stop being received as soon as the XOFF character is sent.

[1] Special thanks to Digi support for digging me out when I got confused over this.

# Appendix C

# Using X-CTU under Linux

Linux being my operating system of choice, I've always felt some frustration at having to dig out a Windows machine when I want to update XBee firmware. I now know that I was wrong in this – there's actually no need at all to run Windows in order to run X-CTU for firmware upgrades.

Google has shown me that I'm far from being the first to discover this, but I thought I'd publish it because I've seen a few questions about it in the XBee forums and I've not yet seen the solution posted.

The key is the `wine` program, and finding out how to make it work for X-CTU.

Wine is available for Linux and for Mac, so in principle this explanation should also apply for Mac users. I haven't been able to try it on a Mac though.

The wine home page is `http://www.winehq.org/`.

So if you use Linux or Mac and want to try this, here's the recipe.

Obviously, make sure you have wine installed. Typing

```
which wine
```

as a command is one way to find out. If it isn't installed, your distribution probably has it available as a package.

Now download the latest version of X-CTU. I got version 5.1.4.1 which came as a file called `40002637 c.exe`. If a later version has been released by the time you read this, use the later version instead.

Run the command

```
wine 40002637_c.exe
```

That installs X-CTU. If it asks questions, take the default answer every time.

The first time you run wine it creates a hidden directory called `.wine` in your home directory. This directory contains some configuration stuff and a subdirectory called `drive_c`. The `drive_c` directory corresponds to the C: drive (surprise surprise).

You'll find the X-CTU installation in `~/.wine/drive_c/ProgramFiles/Digi/XCTU` and if you navigate to there you can run X-CTU with

```
wine X-CTU.exe
```

Try it by all means, but there's more to do before it will run successfully.

Actually on my distribution (Fedora Core 11) I also got a desktop icon for launching X-CTU. Your mileage may vary.

In the `.wine` directory there's also a directory called `dosdevices`. The `dosdevices` directory contains symbolic links. Here's how mine looked originally:

```
[john@spike dosdevices]$ ls -l
total 0
lrwxrwxrwx. 1 john john 10 2010-02-13 18:30 c: -> ../drive_c
lrwxrwxrwx. 1 john john 1 2010-02-13 18:30 z: -> /
```

Now at this point the recipe forks. I'm using an RS-232 development board, so I needed to put into `dosdevices` a symbolic link called `com1` to the serial port device file (`/dev/ttyS0` in my case). To do that from within `dosdevices`:

```
[john@spike dosdevices]$ ln -s /dev/ttyS0 com1
```

Note: no colon after `com1`.

Next check the permissions on `/dev/ttyS0` (or whatever you're using). I found it was owned by root and in the dialout group, with permissions allowing access only by those users. There are two solutions to this: 1. Make it available to all users (you need to be root for this command):

```
chmod o+rw /dev/ttyS0
```

or 2. Add yourself to the dialout group (also as root):

```
usermod -a -G dialout john
```

and (as yourself) log out and in again for the change to take effect. Use the

```
groups
```

command to confirm that you've joined dialout.

If you're using a USB development board, the procedure will be similar but with a twist, because USB `/dev` files are typically created only when a device is plugged in. There will be a different group to join (perhaps uucp), or to change the permissions you have to dig in the `/etc/udev` directory. The device files have names like `/dev/ttyUSB0` so you'll need to set symbolic links accordingly. As you may gather from the slightly vague wording, I haven't actually tried this.

Now you're ready to run X-CTU. Have a development board or equivalent connected at this point, so that there's something for it to see.

Launch X-CTU and the next problem becomes clear: it won't detect any COM ports. Fear not. Click on the User Com Ports tab and enter the port by hand. So if you're using COM1 put a 1 in the Com Port Number box, then click on Add. Now COM1 appears in the Select Com Port box and you can click there to select it. Hit the Test/Query button and if all has gone well you'll get a dialog box with the modem type and its firmware version.

Unfortunately X-CTU won't remember the COM port next time it's run, so you have to do the last step every time. Still, that's not a huge imposition.

And at that point you're ready to go. Click on the Modem Configuration tab and upgrade the firmware on your module.

A few caveats:

1. The display of parameter settings didn't look right to me, and I wouldn't like to use X-CTU in this way for viewing or changing parameters. That's ok by me – I've made my own software for that, and in this exercise I was only interested in firmware upgrades.

2. X-CTU didn't automatically discover the modem type and function set so I had to select those by hand from the menus.

3. It also didn't manage to read the parameter values when I asked it to.

4. The menu for selecting the version of the upgrade didn't appear in the proper order. The latest version was somewhere in the middle.

5. While X-CTU was running, wine was generating various "fixme" type messages to the console. That probably explains the other caveats, but hey – the firmware upgrades worked!

Maybe further work would solve the caveats, but for now I'm happy with what I have. If anyone else tries this, please use the forum to let us all know how you get on.