UNIVERSITI TEKNOLOGI PETRONAS

ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

FINAL YEAR PROJECT II

INTERFACING CMOS CAMERA WITH ARM MICROCONTROLLER FOR SMALL ROBOTIC PLATFORM

Final Report Submitted in Partial Fulfillment of the Requirement for the Bachelor of Engineering (Hons) in Electrical & Electronics Engineering

MOHAMED MEERA HUSSIEN

15093

DR. MOHD ZUKI YUSOFF

CERTIFICATION OF APPROVAL

INTERFACING CMOS CAMERA WITH ARM MICROCONTROLLER FOR SMALL ROBOTIC PLATFORM

by

Mohamed Meera Hussien Bin Jawhar Sathik

15093

An project dissertation submitted to the

Department of Electrical & Electronic Engineering

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONIC ENGINEERING)

Approved by,

____

Dr. Mohd Zuki Yusoff

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or performed by unspecified sources or persons.

Mohamed Meera Hussien Bin Jawhar Sathik

ID: 15093

ABSTRACT

This proposal presents an autonomous color tracking mobile robot with obstacle avoidance. The robot detects the object color through CMUcam2. CMUcam2 is a camera which comes with an onboard image processing capabilities and can be interfaced directly to microcontroller. The robot follows the object according to the algorithm that had been programmed. The heart of the system is Raspberry Pi. Raspberry Pi is a credit-card sized computer that is powered by ARM11. This device runs on Linux OS and can be programmed to do specific functions. It supports many environments such C, Python and many more. The interfacing between the CMUcam2 and Raspberry Pi is performed through serial communication. The robot is programmed with Linux as the OS and C language to do the algorithm. In this project, a simple image processing technique is employed to process the image. The robot should be able to detect a specific color and follow it.

Experiments are conducted in order to be familiar with the tools used in this project. This involves experimenting with the Raspberry Pi and CMUcam2. A simple interfacing and calibration are performed in order to gain a better understanding on the tools used. For the Raspberry Pi, it is interfaced together with Pi Face. A simple programming is done using python and the output is observed. Meanwhile, for the CMUcam2, it is interfaced to the PC and basic calibration is done before taking the picture. Once the individual calibration is successful, both the device are interfaced together to perform the desired function.

ACKNOWLEDGEMENTS

All praises to ALLAH S.W.T, the most merciful and the most gracious. I am able to finish this report, with His help and guidance. I would like to extend my highest gratitude to the people who have helped me in completing this report.

First, of all, I would like to thank Dr. Mohd Zuki, my final year project supervisor, whom has assisted me a lot in my final year project 1. From time to time, he gives advices on how to complete the final year project. He also shares his experiences, which was really beneficial in doing this final year project. Besides that I also would like to thank Mr. Duc Lee who has helped me in doing this project. Whenever I face problem, he will help me to solve the problem. He always has solution for my problem and also motivates me in completing this project.

During this 14 weeks, I also have received good support from the technicians, Mr. Hasrul, Mr. Musa, and Mr. Yasin in building the robot model .Not to forgotten my fellow colleagues, Khairul Ariffin, Alfred, Rasid, and Zulhilmi for their motivation and encouragement they gave me to go further in my life.

Contents

Table of Figures

CHAPTER 1

PROJECT BACKGROUND

1.1 Background study

In today's world robot are used not only in industry but in our daily lives as well. Robotics is becoming more and more important because it makes the human life easier. Robotics is mainly used for military, medical and educational purposes [1]. In education level robotics are used to help student to learn about electronic and mechanical concept.

In the old days the robot are very big in size and takes lot of space, but now it is very small and becoming mobile. Mobile robots are easier to move and can go to the places where it is impossible for human to go. One of the examples is the robot that was send to the moon known as Lunar. Mobile robots are usually autonomous. Mobile robot sense the outside world with its sensor. Previously the type of sensors used is most likely mechanical and electronic sensors such as tactile switch, motion sensor and temperature sensor. This makes the mobile robot to be smart, but not smart enough to recognize the surroundings. Researches integrate image processing to mobile robots which makes it to become smarter. The robot can have a better sense and the decision made by using image is much more accurate in comparison to the input from passive sensor.

This project is built upon Raspberry Pi and CMUcam as the core of the robot. Raspberry Pi is the newly developed device. It is a powerful device, whereby it can run an OS on the board itself. This device is cheap considering the function it offers to user. Moreover it can be programmed to do a specific function. CMUcam is a camera which comes with a processor on its board. This device is powerful that, it can do basic image processing by itself, without requiring additional circuitry. By sending certain command this can be achieved. Also it can be interface to other controller such as microcontroller through serial communication. On the other side there are some constraints in using the Raspberry Pi. The operating voltage of

Raspberry Pi is 3.3v, hence there will be some difficulties to control motors such DC motor or servo motor. Moreover a special kind of connector is required to interface with any other devices. And lastly in order to program it, it requires having a monitor, keyboard and mouse.

This robot can be a good learning tool to study robotics since it involves lots of engineering knowledge such programming, image processing and electronic. And also it can be further developed to do a specific task such for firefighting and much other application.

1.2 Problem Statement

In developing this project, there are few grey areas that need to be overcome. The first problem is to make the robot to work as close as to real-time. Since there is image processing involved, there may be some delay in the decision making of the robot. There a lot of image processing available. A good method should be implemented in order to ensure that the image processing does not take much time and it is able to work in RTOS. The method chosen should be able to detect the color of object. As this is the main objective of the robot.

The second obstacle is programming the Raspberry Pi. This is because the Raspberry Pi is still new and not many people have tried to interface it with other devices, in this case to interface the Raspberry Pi with CMUcam. Theoretically, the interfacing is performed through serial communication and the CMUcam should respond to the command send by Raspberry Pi.

The other obstacle is to choose a good platform for the mobile robot. The platform needs not to be too big or too small. It should be just nice to hold all the necessary devices and can maneuver smoothly. Choosing a bad platform will end up in mobile robot that cant maneuver as supposed. Moreover the type of material used to build the chassis of the robot is also important.

Most importantly, the cost of this robot should be low compared to the available robot in the market that offers the same functionality.

1.3 Objective

The objective of this project is to build a low-cost small mobile robot based on Raspberry Pi and CMUcam2. There are two main function of this robot. First it can detect red color object and follow that object. The second is it can avoid obstacles, while following the object.

1.4 Scope of Study

This project focuses mainly in detecting object color and track the object. This includes

- Analyze the image frame and extract the object
- Filtering the image
- Response toward the obstacle detected
- Algorithm to combine object color detection and obstacle detection

CHAPTER 2

LITERATURE REVIEW

2.1 Image processing on Robot

Mobile robots are widely used in a lot of field this includes military, medical, space missions and many more. Vision gives rich information to the mobile robot of the outside world that helps the robot to detect color, object and other properties [2, 3]. There are a lot of cameras that can be used as a vision for mobile robots. One of them is the CMUcam[4].

The vision system of a robotic is used to do a specific function such as to follow line, to detect object, for navigation and many more [2, 5, 6]. In each of the research a specific technique is chosen to process the image. The image will be processed to extract the required data [7]. There are a lot of image processing methods available to process the image taken such as Unscented Kalman Filter and Gaussian [3, 6]. In one of the research conducted on line following using vision, a method called Lane Boundary Pixel Extractor (LBPE) and Hough Transform[[2]] were used in coordinating the image to the frame and translate it to the robotic coordinates using photogrammetric techniques.

The other method used in mobile robots is YCbCr, which is commonly used in color space. It detect the object by the obtaining the threshold of a certain color. The other method that is commonly used is the segmentation method [5]. In this project, a simple method has been employed for detect the object color by using the built-in function of CMUcam, which is screen through all the pixel of image and get the proper pixel value. Once it is obtained the robot then can follow that object, and at the same time doing the screening of the image.

2.2 Real-Time Operating System (RTOS)

In the old day's real time system were meant for critical application only. As the technology progresses and with the invention of mobile phones, PDA's and other handheld devices the RTOS become more important. All this devices are driven by RTOS [8]. RTOS are used largely in embedded system.

There are a lot of different types of RTOS available. Some of the famously used RTOS is VxWorks, Windows CE and QNX. One of the well-known RTOS is Linux, which is a general purpose operating system [9]. Some of the other available RTOSes are μITRON, EmbOS, XMK OS, μTKernel, Echidna, KeilOS, μC-OSII, FreeRTOS, Salvo, Erika, SharcOS, TinyOS, Hartik and PortOS. All most all of the RTOS can be programmed in C language. Each of this RTOS holds different characteristics in terms of system management, Interrupt management, memory management and many more [10]. In ref [10] they compared all the available RTOSes, and found that each of the RTOS have their own advantages and disadvantages. For instance uC-OSII has lower code size(ROM) compared to uTKernel but it has higher capacity in Data size (RAM).

Linux are commonly used in most of the RTOS especially in embedded operating system. Due to its uniqueness it has become the world second most used operating system. The advantage of Linux is that it is open source and has a lot of software resources, and supports multithread, user, and process. Moreover it offers great portability, functions and stability [11]. Hence this makes Linux to stand out among the other OS. The reason to choose Linux OS is because it is capable to manage the processing and work in real-time.

2.3 Decision Making Techniques

The decision making is an important task in mobile robots. The decision which is an output should be able to solve the problem that the mobile robot faced. One of the techniques is the Landmark-based localization techniques [11]. The other decision making techniques are by using by probabilistic mathematical model. This includes Markov and Monte Carlo technique [11]. The other technique is known as novelty detection [12]. In this technique, the robot detects and responds to something that it is not supposed to respond to. The other method is by classification. In reference [13] classification techniques was used to detect object. The output of classification technique depends on the data that was presented to it.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology and Project Activities

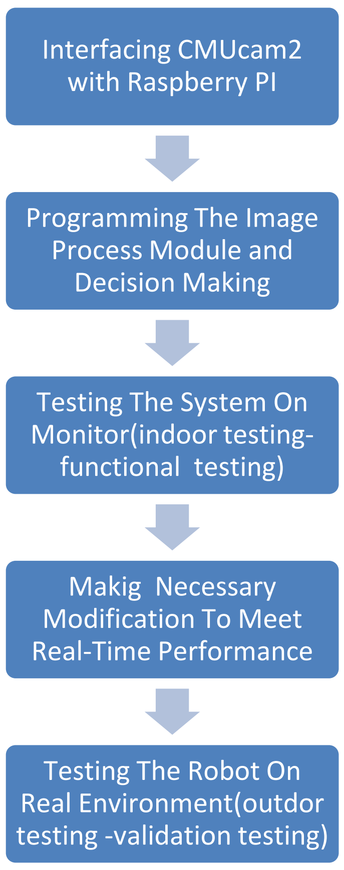

Interfacing CMUcam2 with Raspberry PI
Programming The Image Process Module and Decision Making
Testing The System On Monitor(indoor testing-functional testing)
Makig Necessary Modification To Meet Real-Time Performance
Testing The Robot On Real Environment(outdor testing -validation testing)

Figure 1 Project Flow

The project flow is as shown in the figure above. It involves five processes. Each of this process is crucial in order to ensure that the mobile robot can work in real-time system. The first process is to interface Raspberry Pi with CMUcam2 through serial communication. Once the interfacing is successful, the next process is to programme the Raspberry Pi with image process module and decision making. The third process is to test the system on-monitor. In this test, the system will be calibrated and tested at different condition of lighting and the response of the system is recorded. In the fourth process, necessary modification is performed based on the data collected in the previous process. And finally the robot is tested in real-time environment.

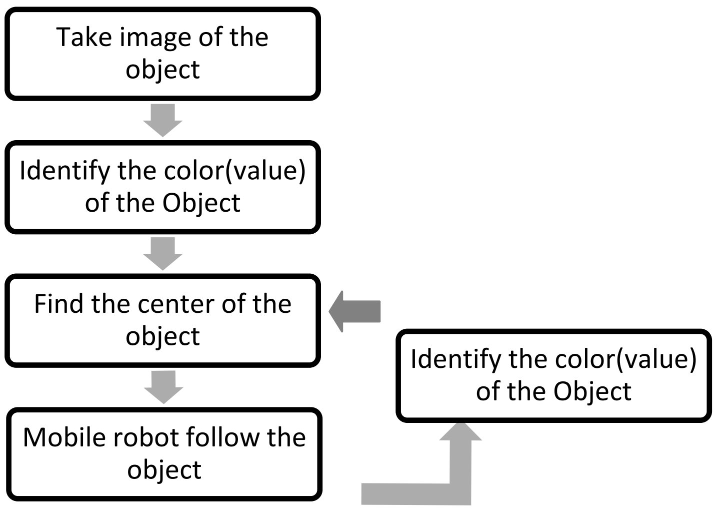

Take image of the object
Identify the color(value) of the Object
Find the center of the object
Mobile robot follow the object
Identify the color(value) of the Object

Figure 2 Image Detection Flow

The method used to track the object is first by capturing the image and transmit it to Raspberry Pi (Microprocessor). This is performed by sending certain commands to CMUcam2 to capture image. Once the image is obtained, it is processed and analyzed to see if the color of the object matches the color of what had been programmed. If yes, then the center of the image is obtained for tracking. Basically, the area that the object could be is first selected and the color of that zone is obtained once the object is found within the zone. The CMUcam sends the centered of the object found in x and y coordinates [14]. Then the Raspberry Pi will send signals to the servo motor to move towards the object. The mobile robot will maneuver by following the object detected.

3.2 Experimental Procedures/Approach

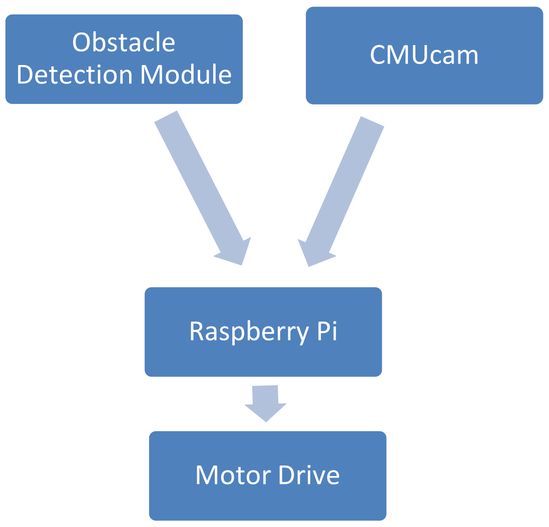


Figure 3 Project Activity

The project flow begins with the detection of image and also obstacle detection. Once the data is obtained, it is send to Raspberry Pi to process the data. The Raspberry will then send command to motor either to make one of the six movement. There is few experimental procedures involved in this project. First is programming the Raspberry Pi, followed by testing the CMUcam under different condition of lightning and different contrast and lastly interfacing the devices

3.2.1 Programming Raspberry Pi

Firstly, Raspberry Pi needs to be programmed in order to be able to interface with CMUcam or other devices. Ample of time need to be spend in testing the device and do simple task to get familiarize with embedded OS programming. The programming is performed in Linux OS environment using C language. All the algorithms of the robot is programmed in C. Moreover this device is considered still new in the market, hence will require extensive test on the board in order to ensure that it is really suitable to be used in this project.

3.2.2 CMUcam2

CMUcam2 have various built in functions. Each of the functions needs to be tested separately before interface with Raspberry PI. This is because images taken under good lightning and poor lightning will not be the same. Hence obtaining the image data under different lightning conditions will be helpful when it comes to do the algorithm to detect the color of the object. Another thing is the different of contrast of the color. This is because an object that is observed under sunlight and under a fluorescent lamp will definitely will have differences.

3.2.3 Mobil robot movement

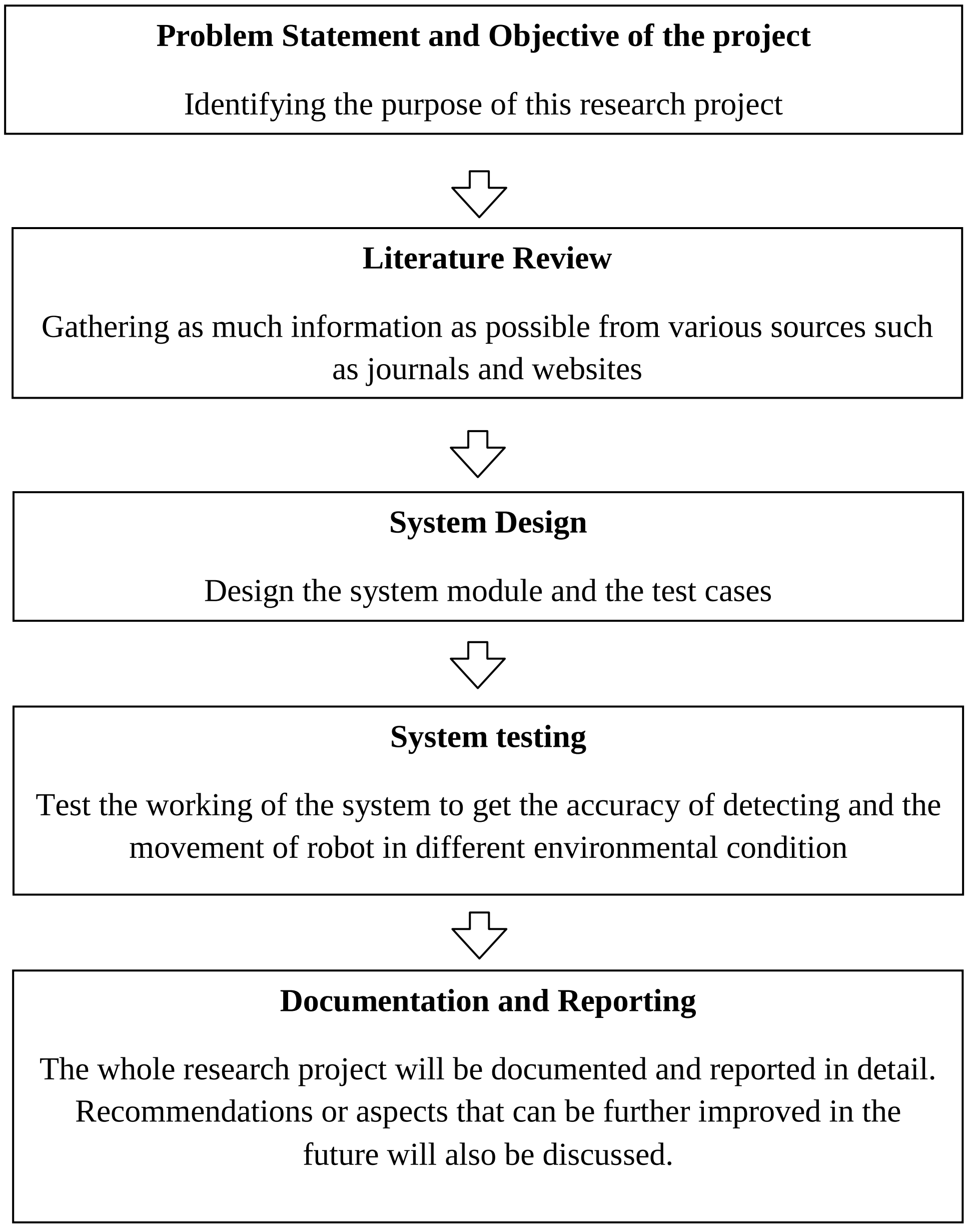
The movement of the robot is about how the robot will respond to the image and obstacle detected by the robot. The robot will need to decide either to move forward, backward, left or right depend on the object that is being tracked. An algorithm should be created to ensure that the mobile robot is able to move according to the object movement. Together with object color detection the robot will also be able to detect obstacles. Altogether there are 6 movements which are forward, backward, left, right, slightly left and slightly right.

3.2.3 Interfacing

All the system will be integrated together to perform the specific task. Need to ensure the system works as it supposed after the interfacing. Usually all the devices can work when tested separately, but when interfaced together, some problems may arise. Need to come up with plans to mitigate the problems.

3.3 Key Milestones

Several key milestones for this research project must be achieved in order to meet the objective of this project. It begins from identifying the problem statement and objective of the project. The second milestone is gather information related to this project from various sources. Based on the information collected, a system is designed with a few test cases. Once the system is in place, it is tested to get the accuracy of detecting and movement of robot in different environmental condition. And finally all the research and works performed is documented.



Problem Statement and Objective of the project

Identifying the purpose of this research project

Literature Review

Gathering as much information as possible from various sources such as journals and websites

System Design

Design the system module and the test cases

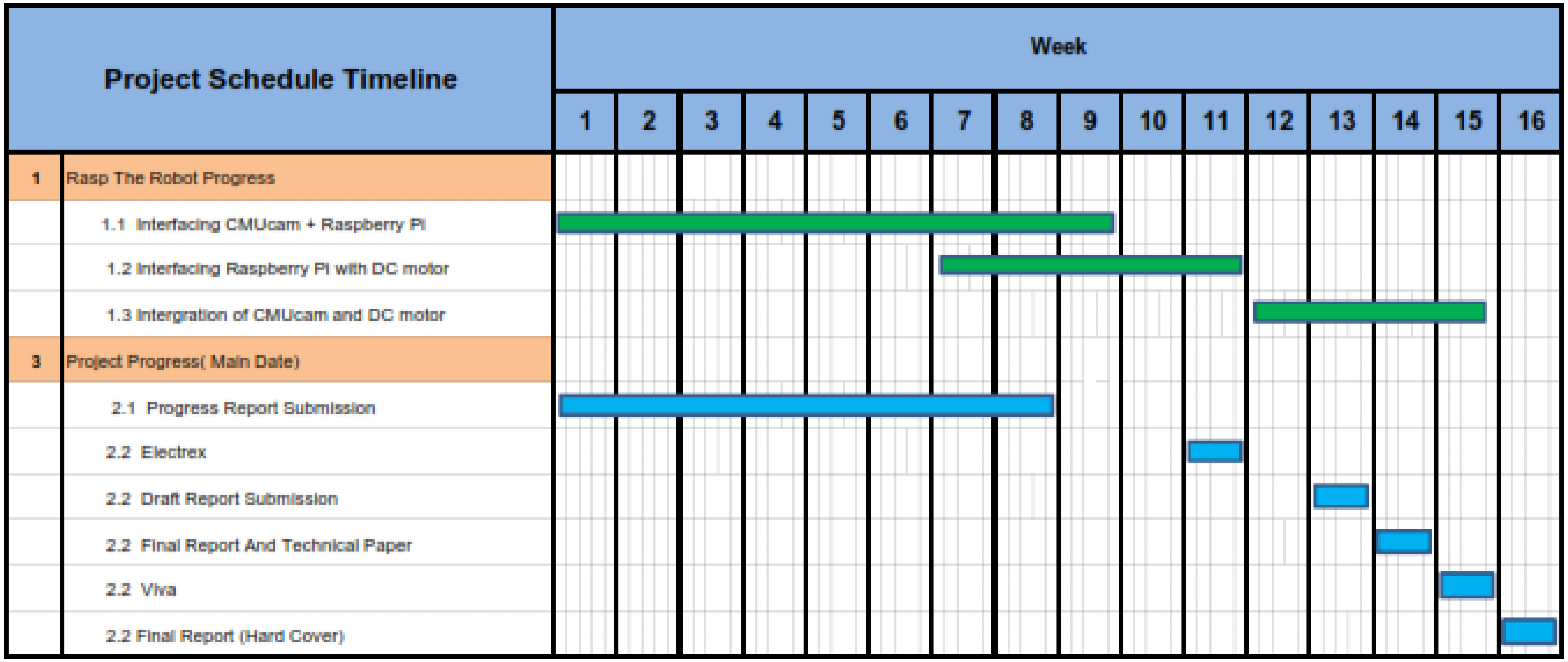
System testing

Test the working of the system to get the accuracy of detecting and the movement of robot in different environmental condition

Documentation and Reporting

The whole research project will be documented and reported in detail. Recommendations or aspects that can be further improved in the future will also be discussed.

3.4 Gantt Chart



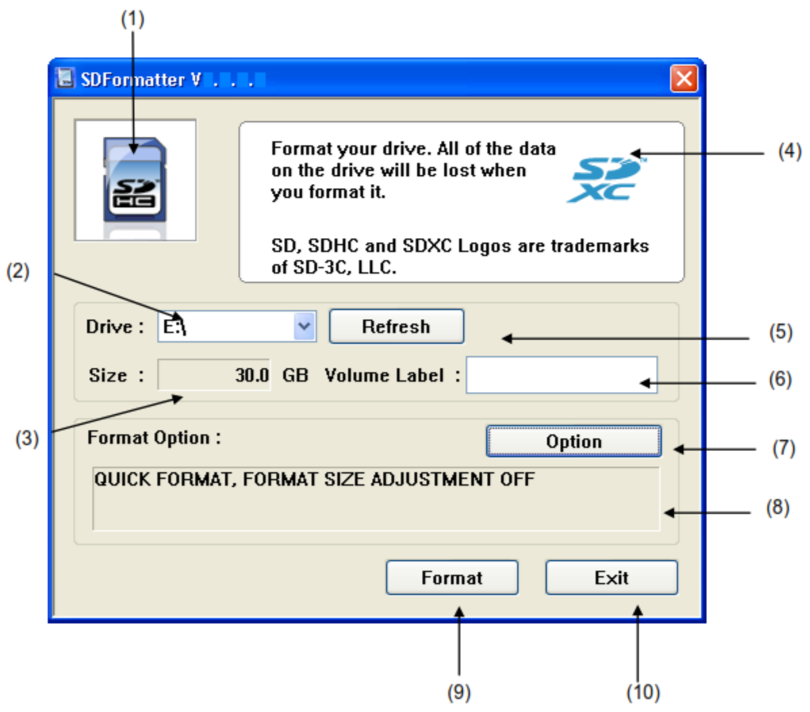
CHAPTER 4

RESULTS & DISCUSSION

4.1 Setting up The Raspberry PI

Raspberry Pi runs on a operating system (OS). To date there are a few OS that are available in the market that can be used to run the raspberry pi. These OS has advantage and disadvantages one another. To name some of the operating system is Pidora, Archlinux, Raspbian, OpenELEC and RiscOS. In this project, Raspbian OS is chosen as the OS to work with. The advantage of using the Raspbian is that, it runs on Linux platform and based on Debian optimized for the Raspberry Pi hardware.

The OS need to be installed in a SD Card before it can be used in Raspberry Pi. In order to this, a memory card with a minimum size of 4 GB is required. I order to format the memory card a software is used which is known as SD Formatter.. The figure below shows the GUI of the application and some basic description of the software.



Format your drive. All of the data on the drive will be lost when you format it.

SD, SDHC and SDXC Logos are trademarks of SD-3C, LLC.

Drive : E:\

Size : 30.0 GB

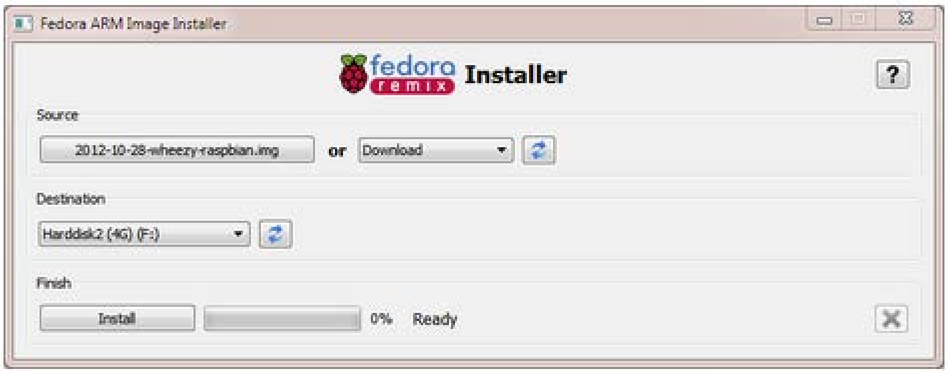
Format Option :

QUICK FORMAT, FORMAT SIZE ADJUSTMENT OFF

Figure 4 SD Formatter Window

1) Status – to indicate the status
   i. Blue: They are functional
   ii. No Logo: Interface device is disabled
   iii. Gray – Card is not recognized

Once the card is formatted, the OS can then be installed. In order to install the OS a special software is used which is known as "Fedora Arm Installer". The screenshot of the software is as shown below


Fedora ARM Image Installer

fedora remix Installer

Source

2012-10-28-wheezy-raspbian.img or Download

Destination

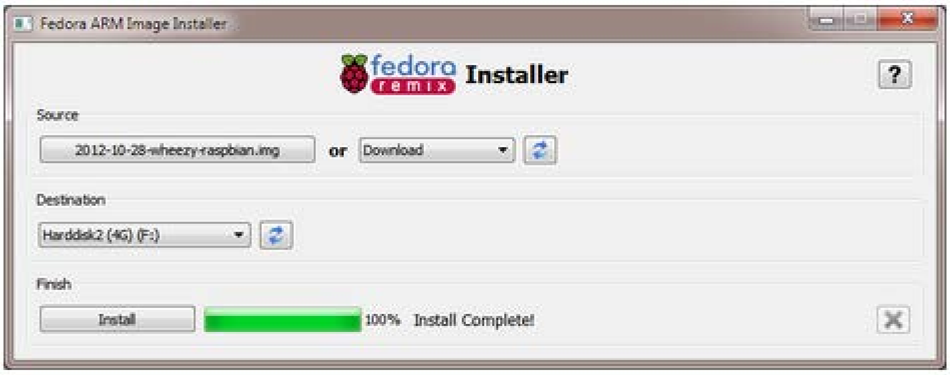
Harddisk2 (4G) (F:)

Finish

Install 0% Ready

Figure 5 Fedora ARM Installer

Once we choose the source (the OS), we can download it to the SD card that had been formatted earlier. Once it has successfully downloaded the OS content, the window will show "Install Complete".


Fedora ARM Image Installer

fedora remix Installer

Source

2012-10-28-wheezy-raspbian.img or Download

Destination

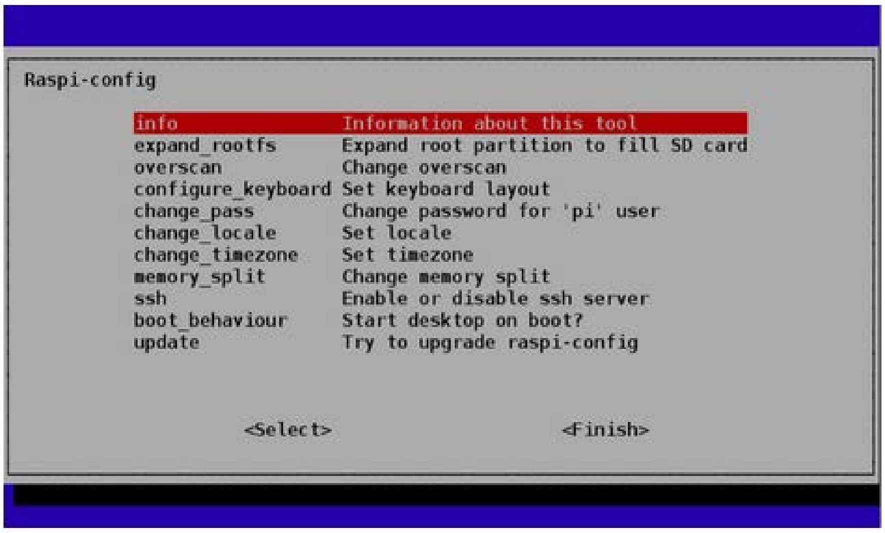
Harddisk2 (4G) (F:)

Finish

Install 100% Install Complete!

Figure 6 Fedora ARM Complete Installation on the SD Card

Now the SD Card can be inserted into the Raspberry and can power up. On the first run, a window will pop up named "Raspi-Config". In this window, the settings of the raspberry pi can be configured.



```
Raspi-config

info                Information about this tool
expand_rootfs       Expand root partition to fill SD card
overscan            Change overscan
configure_keyboard  Set keyboard layout
change_pass         Change password for 'pi' user
change_locale       Set locale
change_timezone     Set timezone
memory_split        Change memory split
ssh                 Enable or disable ssh server
boot_behaviour      Start desktop on boot?
update              Try to upgrade raspi-config

            <Select>                    <Finish>
```

Figure 7 Raspi – Config Window

There are many settings available for the user to choose from. Once the setting is done, the Raspbian is ready to go. On the click of finish, a new window will appear which looks like as shown in figure below.

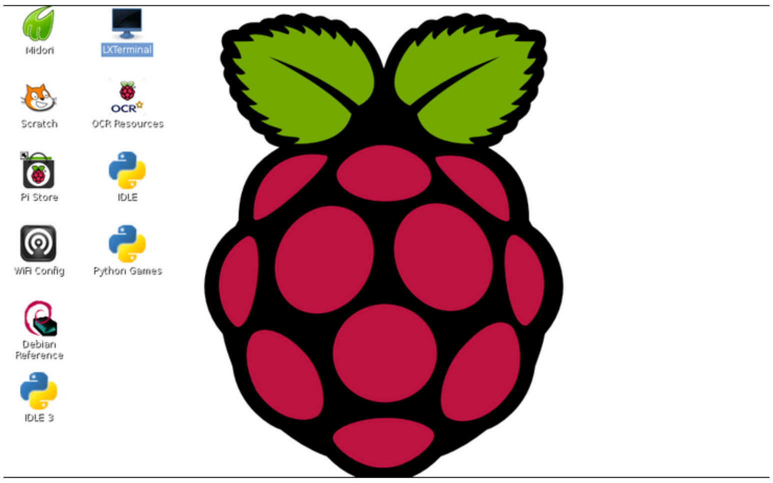


Figure 8 Raspbian Window

This is the window whereby it will allow the user to do programming and also allow the user to make the raspberry pi to communicate with the outside world. In this window the user will be able to make connection to the internet, do python programming and other basic computer task.

4.2 Communicating the raspberry pi with the outside world

There are many methods that can be used to communicate the raspberry Pi to the outside world. One such way is to interface the Raspberry Pi with Pi Face. Pi Face is a device that can be connected directly to the Raspberry Pi before being interfaced to the outside world. The figure below illustrates the Pi Face.

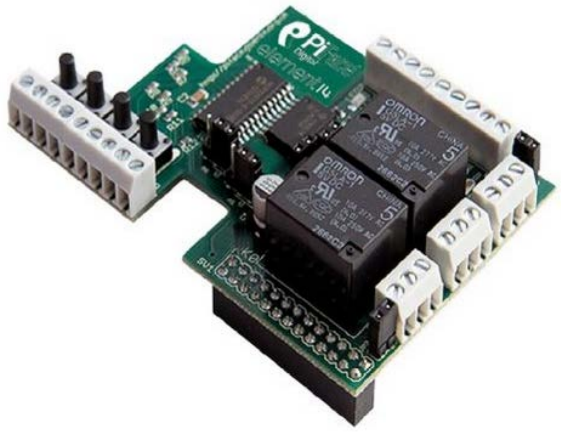


Figure 9 Pi Face

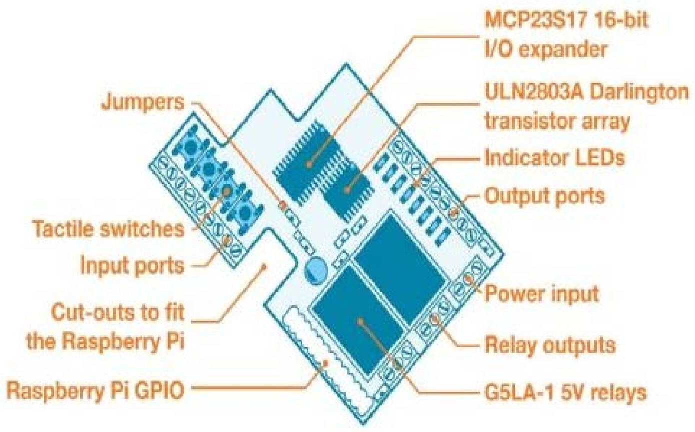
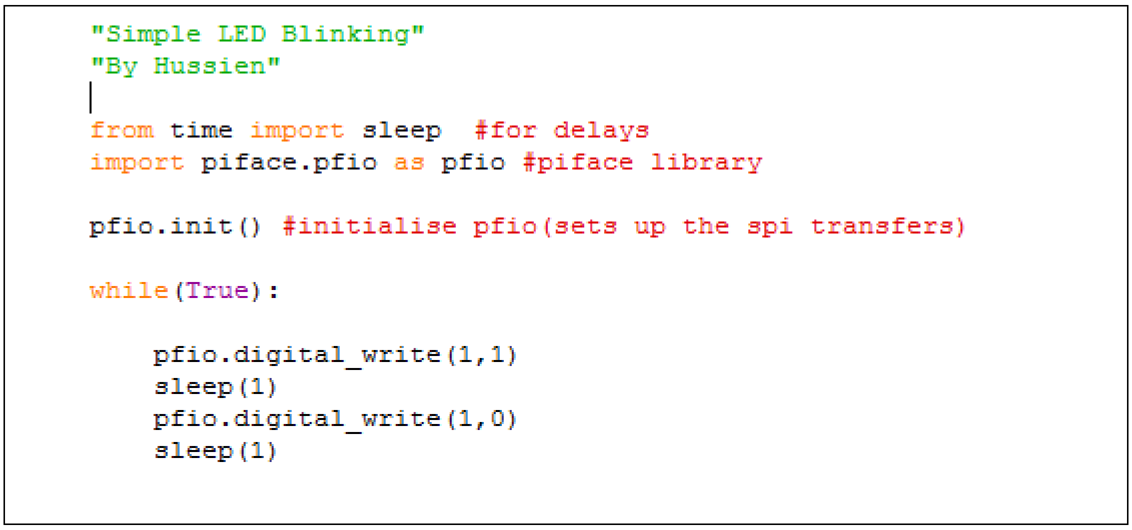


Figure 10 Pi Face

Basically the Pi Face provide digital input/output interface for the Raspberry Pi. In the first stage, a simple LED blinking program is tested. A simple program is written base on Python programming language. It is explained more briefly in the next section.

4.2.1 Simple Led Blinking



```
"Simple LED Blinking"
"By Hussien"

from time import sleep  #for delays
import piface.pfio as pfio #piface library

pfio.init() #initialise pfio(sets up the spi transfers)

while(True):

    pfio.digital_write(1,1)
    sleep(1)
    pfio.digital_write(1,0)
    sleep(1)
```

Figure 11 LED Blink Program

In the program above, the LED at pin 1 blinks every seconds. The actual image is as shown in the figure below

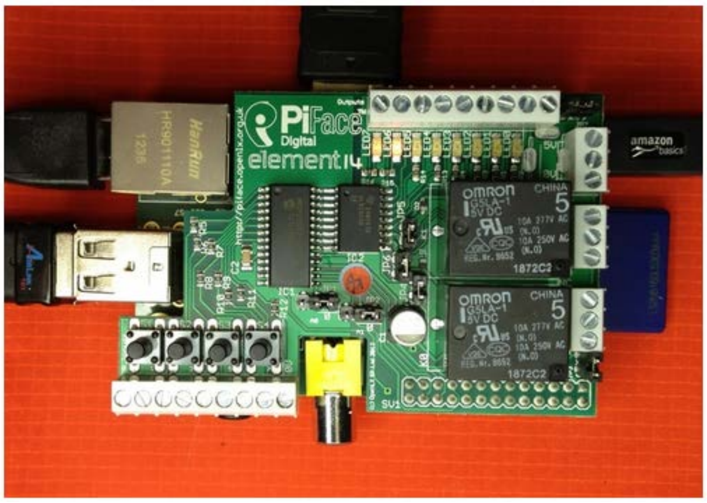
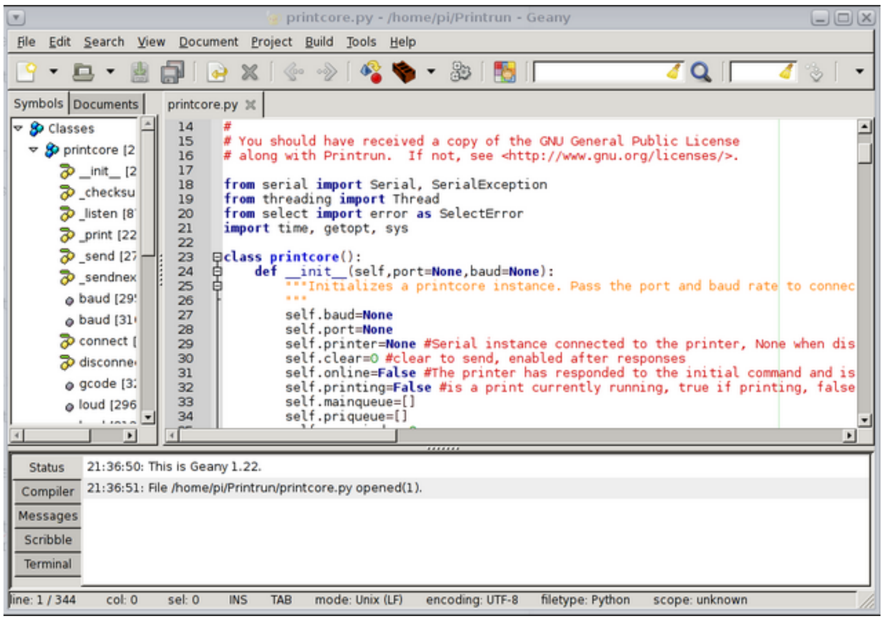


Figure 12 Interfacing with Raspberry Pi and Pi Face

Moving forward, a different approach is used to interface the Raspberry Pi to the outside world. Whereby rather than using the Pi Face, the Raspberry is directly interface through the general purpose input/output (GPIO) pins. In order for this to took place, a different IDE is used which is called “Geany”. This Geany uses C language and it allows the program to control the GPIO pins. In order to be able to control the GPIO pins, a special C library is required known as BCM2835. This library provides access to GPIO and other functions of the Broadcom 2835 chip. Whereby it allow the user to access the GPIO pins on the 26 pin of on the Raspberry

Pi to control and interface with various external devices. It also provides function for reading digital inputs and setting digital outputs, using SPI and I2C and also to access the system timers. Moreover pin detection is supported by the polling method. Each of these pins has a specific function that has been mentioned above. Below is the screenshot of Geany



```
#
# You should have received a copy of the GNU General Public License
# along with Printrun.  If not, see <http://www.gnu.org/licenses/>.

from serial import Serial, SerialException
from threading import Thread
from select import error as SelectError
import time, getopt, sys

class printcore():
    def __init__(self,port=None,baud=None):
        """Initializes a printcore instance. Pass the port and baud rate to connec
        """
        self.baud=None
        self.port=None
        self.printer=None #Serial instance connected to the printer, None when dis
        self.clear=0 #clear to send, enabled after responses
        self.online=False #The printer has responded to the initial command and is
        self.printing=False #is a print currently running, true if printing, false
        self.mainqueue=[]
        self.priqueue=[]
```

```
21:36:50: This is Geany 1.22.
21:36:51: File /home/pi/Printrun/printcore.py opened(1).
```

```
line: 1 / 344    col: 0    sel: 0    INS    TAB    mode: Unix (LF)    encoding: UTF-8    filetype: Python    scope: unknown
```

Figure 13 Geany Screenshot

For the initial stage a simple led blinking code is developed to be familiar with Geany tool. The led blinking program is as shown below

```
#include <bcm2835.h>

// Blinks on RPi pin GPIO 11
#define PIN RPI_GPIO_P1_11

int main(int argc, char **argv)
{
    if (!bcm2835_init())
        return 1;

    // Set the pin to be an output
    bcm2835_gpio_fsel(PIN, BCM2835_GPIO_FSEL_OUTP);

    // Blink
    while (1)
    {
        // Turn it on
        bcm2835_gpio_write(PIN, HIGH);

        // wait a bit
        delay(500);

        // turn it off
        bcm2835_gpio_write(PIN, LOW);

        // wait a bit
        delay(500);
    }
    return 0;
}
```

Figure 14 Led Blinking Program

The code is then compiled using Geany. The another way to compile the code is by using the make file. This is because what Geany does is, it compiles files not projects, hence to solve this a make file is required. A simple make file would be something shown in figure below.

```
all: output_file_name

output_file_name: main.o
        gcc main.o -lbcm2835 -o output_file_name

main.o: main.c
        gcc -c main.c

clean:
        rm -rf *o output_file_name
```

Figure 15 A simple makefile

The make file above is for the LED blinking program. Once the program is compiled, it is tested. The result is the LED starts to blink. Figure below shows the working image of the test.

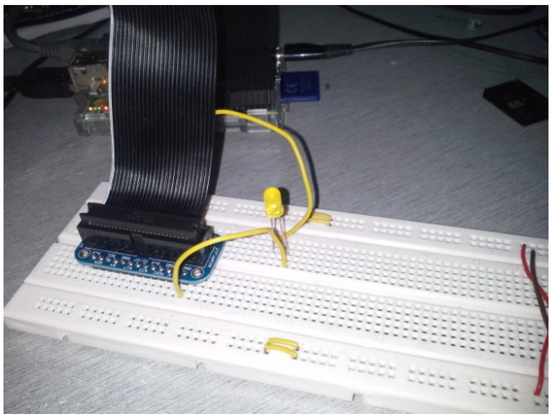


Figure 16 Blinking LED

4.2.2 Serial Communication

The serial communication in Raspberry Pi is done by utilizing the UART pin available on the GPIO Pi on pin 14 and on pin 15 as shown in figure below.

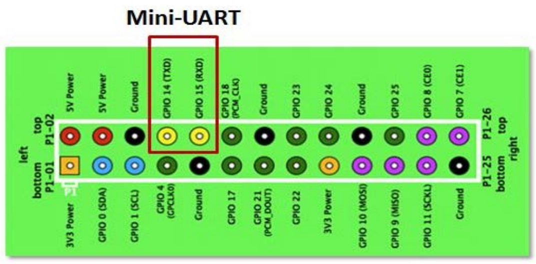


Figure 17 Raspberry Pi UART Pin

In order to access the UART of the raspberry pi, it need to set-up first. There are two ways to access the UART, first is by using third party software and the other method is by using the C code. The task that needs to be done for utilizing the UART through the C code is as shown below. Once the process is completed, it is ready to be used as serial port.

A simple C is developed in order to test the serial port using Geany. There were few errors when the program is compiled. For the moment, the author is trying to debug the code to see what is causing the problem. The sample of the code is included in the Appendices

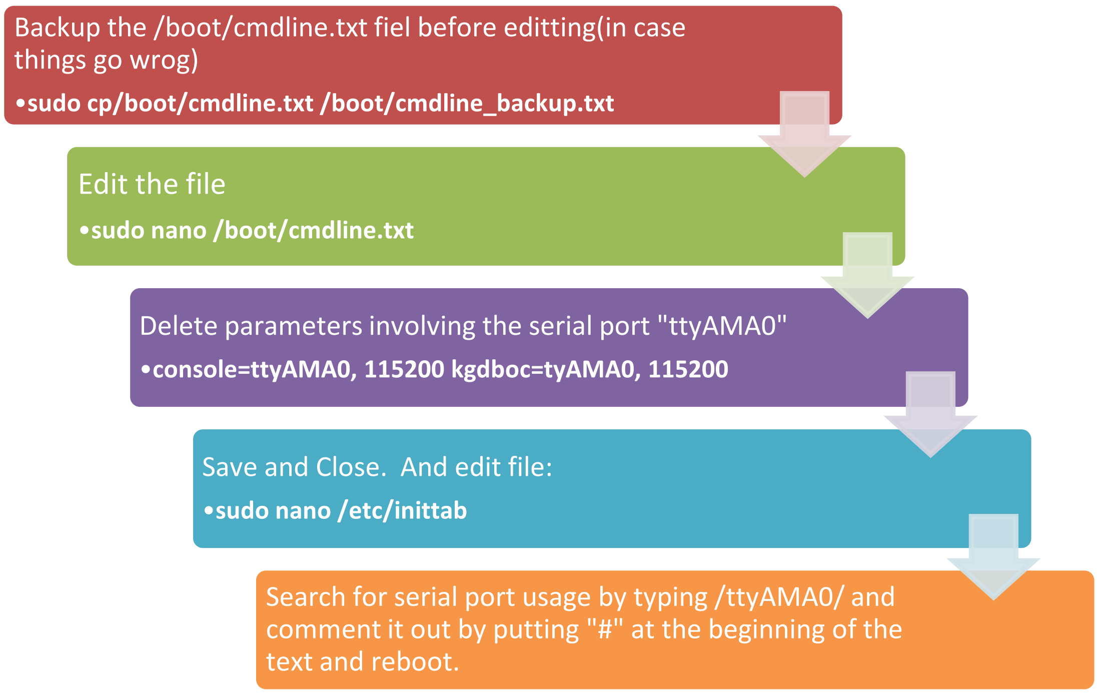
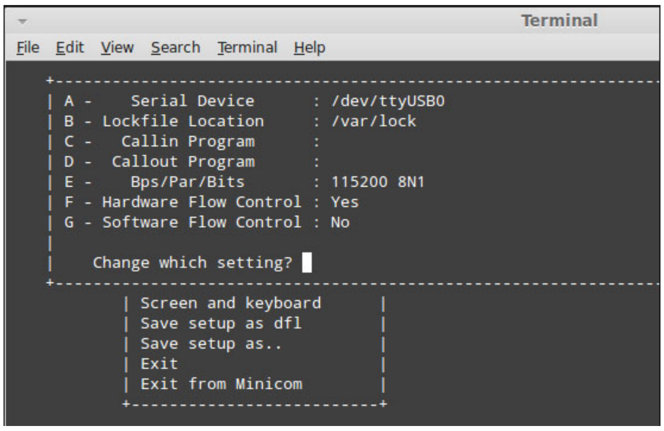

Backup the /boot/cmdline.txt fiel before editting(in case things go wrog)
•sudo cp/boot/cmdline.txt /boot/cmdline_backup.txt
Edit the file
•sudo nano /boot/cmdline.txt
Delete parameters involving the serial port "ttyAMA0"
•console=ttyAMA0, 115200 kgdboc=tyAMA0, 115200
Save and Close. And edit file:
•sudo nano /etc/inittab
Search for serial port usage by typing /ttyAMA0/ and comment it out by putting "#" at the beginning of the text and reboot.

Figure 18 Raspberry Pi Serial Port Setting Steps

The next method is by using the third party software, which known as minicom. The process to set-up is as the same as for the C code. In order to run the minicom, it needs to be installed first into the Raspberry Pi by typing sudo apt-get install minicom. The run the minicom, minicom -b 9600 -o -D /dev/ttyAMA0 need to be type in the terminal window. A new window will pop-up as shown in figure below.



```
 +-----------------------------------------------------------------
 | A -    Serial Device      : /dev/ttyUSB0
 | B - Lockfile Location     : /var/lock
 | C -   Callin Program      :
 | D -  Callout Program      :
 | E -    Bps/Par/Bits       : 115200 8N1
 | F - Hardware Flow Control : Yes
 | G - Software Flow Control : No
 |
 |    Change which setting?
 +-----------------------------------------------------------------
         | Screen and keyboard      |
         | Save setup as dfl        |
         | Save setup as..          |
         | Exit                     |
         | Exit from Minicom        |
         +--------------------------+
```

Figure 19 Raspberry Pi Minicom Window

In this window, the setting of the serial port is configured. The most important of all the settings is A - Serial Device, E – Bps/Par/Bits, F – Hardware Flow Control and G – Software Flow control. The settings of the serial port should be as shown in figure above. For the testing purpose, the serial port from Raspberry Pi is connected to a PC. In order to establish this, first of all a circuit need to be build to convert the low-level logic voltage from the UART into the correct voltage levels. This is because the Raspberry Pi runs on 3.3v meanwhile RS-232 connection could have up to a potential 15V running across it. Moreover an UART requires a logic shifter in order for it to function as a serial port. As mentioned above, Raspberry Pi runs on 3.3V, hence the logic shifter will be useful to drive a higher voltage circuit. The logic shifter works by taking a given signal on the 3.3V logic side which is TTL and converts it up to the correct voltage level but at the same time keeping the same logic level.

For this project MAX3232 is used as the logic-level shifter. The reason for choose this is because it can powered via 3.3V and also draws very low current. The figure below shows the pin-out of MAX3232.

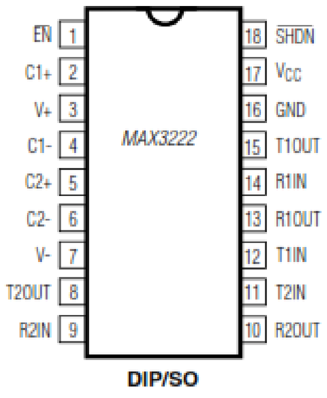


Figure 20 MAX3232 Pinout

The circuit for the serial communication is as shown below. And also the PCB layout of it.

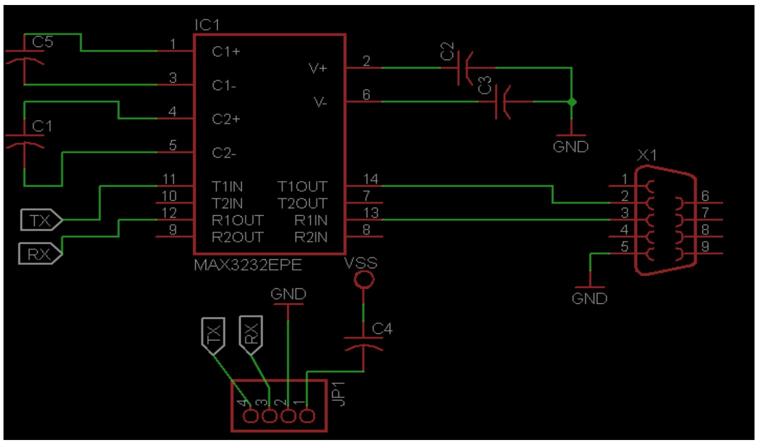


Figure 21 Serial Communication Circuit

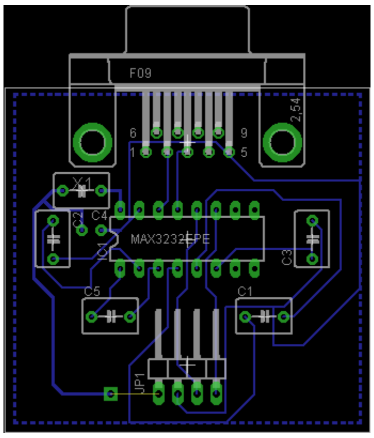
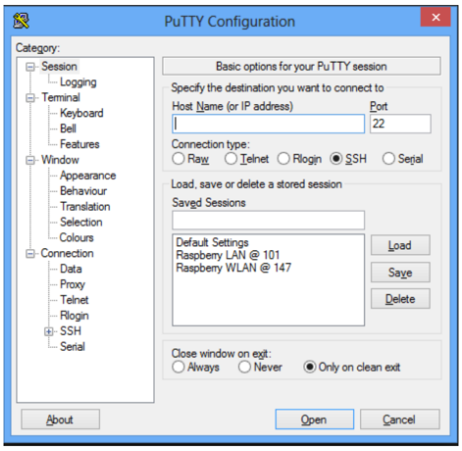


Figure 22 Serial Communication PCB Layout

Once the circuit is in place, the serial communication is tested. In order to test the communication, software called PuTTy is required. The PuTTy is free and a open source terminal. It enables the serial communication to be tested. The appropriate port is selected and it is tested. Below is the screenshot of the PuTTy software.


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): | Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

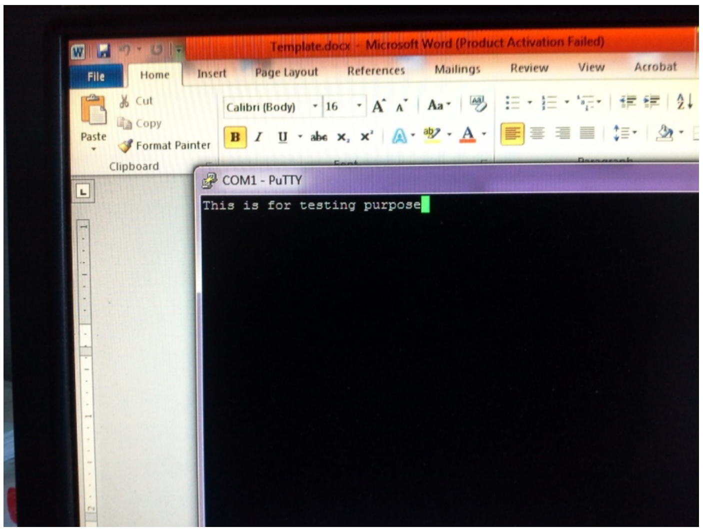
Saved Sessions

Default Settings
Raspberry LAN @ 101
Raspberry WLAN @ 147

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Figure 23 PuTTy Softare

Once the setting is done, the communication is tested. The result is, the two devices can communicate between each other. The characters type in one end appears on the other end of the device. Image below shows the image of the communication and also the circuit of the serial communication.



```
This is for testing purpose
```

Figure 24 PuTTY received file from Raspberry Pi. The text shows "This is for testing purpose"

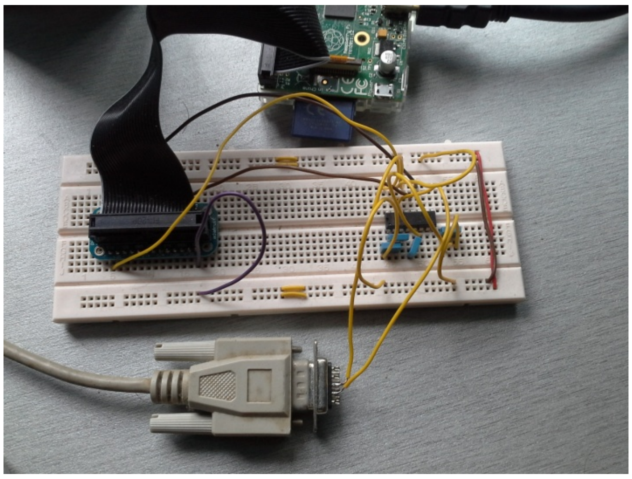


Figure 25 Serial Communication Circuit

4.3 CMUcam

The next main device that is investigated is that, the CMUcam2. The CMUcam2 is a camera module that is able to capture picture and do basic image processing functions. By interfacing the CMUcam2 with PC, some pictures are taken. Below are the sample images. Also some basic calibration is done on controlling the servo motor directly from the CMUcam2. The movement of servo is observed and cross checked with the given input.

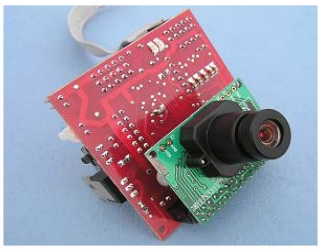


Figure 26 CMUcam2

The camera is interfaced to the PC via a serial port. By sending certain command, the camera is able to capture picture. Below is the sample picture captured using CMUcam2. The captured image can then be processed to obtain specific information from the captured image. In this project, the centre of the image is obtain and is refreshed every few seconds. In order to test the function of the camera, the CMUcam2 GUI can be used to test the camera and the take sample pictures. The sample pictures taken are shown in figure 28 and figure 29. The CMUcam also has the capability to interface with servo motor directly. It can interface to up to 5 servos at one time. For the testing purpose, the CMUcam is attached to the servo motor and controlled through the serial port. The testing is done on the pan and tilt movement of the servo motor. It is observed that, the CMUcam sends certain signal to the servo motor in order for the camera to be on track on to the color detected. Even though the movement of the camera to the image is not that smooth, but the camera is still able to track the color of the object. The image below illustrates how the camera is attached to the servo motor.

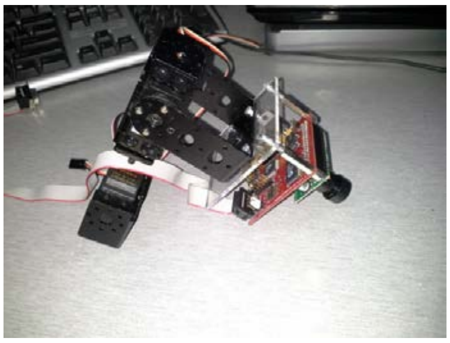
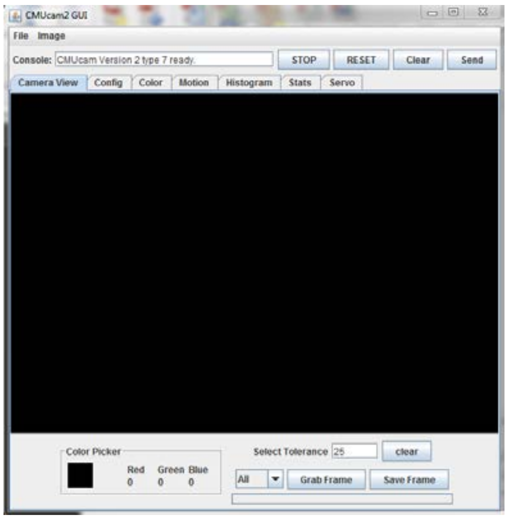


Figure 27 CMUcam Attached o Servo Motor



Console: CMUcam Version 2 type 7 ready.

Red 0 Green 0 Blue 0

Select Tolerance 25

All

Figure 28 CMUCam2 GUI

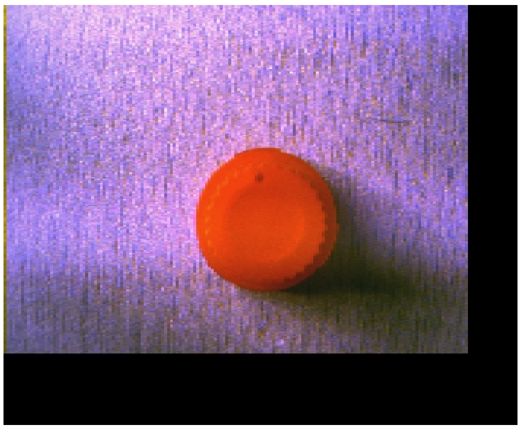


Figure 29 Image 1 Taken Using CMUcam2

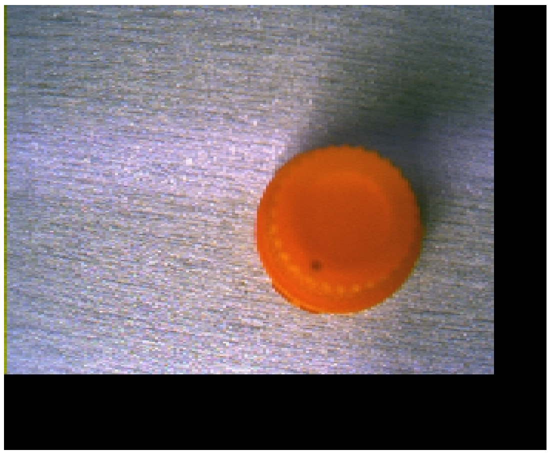


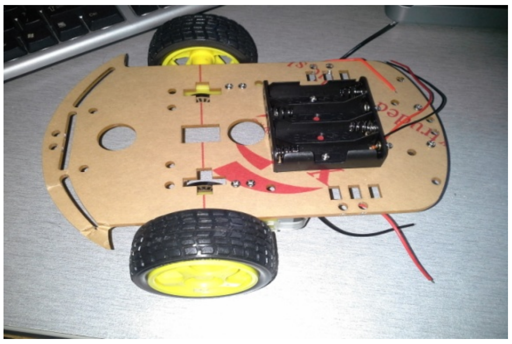
Figure 30 Image 2 Taken Using CMUcam2

The image shown in figure 29 and figure 30 are sample of image taken using the CMUcam2. The main purpose of capturing the red color object is to observe the difference value of captured object. Both the images are taken under different lighting condition. The maximum and minimum value for red color is seemed to be different in both the conditions. Hence this value will be crucial in maneuvering the robot to the right position.

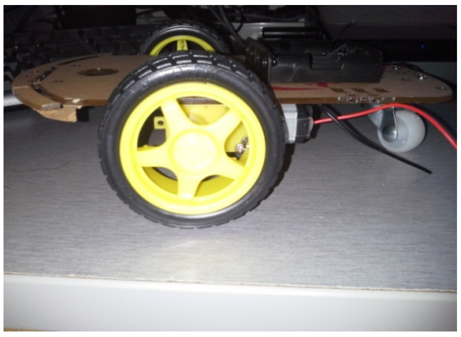
4.4 Robot Platform

The robot platform used in this project is a mobile robot. It has a dimension of the robot is 22 x 17.5 cm. The robot has two DC motors that are attached to the tire. And it has a roller for balancing of the robot. It also comes with an AA size battery holder to power up the robot. The space at the top of the robot will be utilized to put the Raspberry Pi and CMUcam.

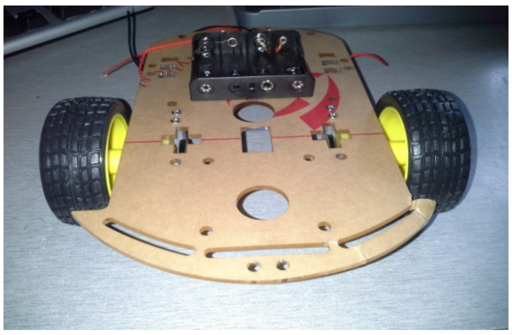
The image below illustrates the robot.



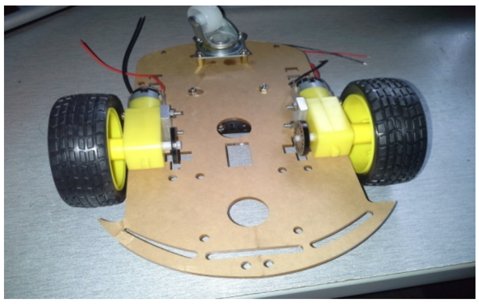
(a)



(b)



(c)



(d)

Figure 31 Image (a) to (d) shows different angle of the robot

4.5 Motor Driver

In order to drive the DC motor, a simple motor driver is employed. This driver is built upon L293D. L293D is a 16-pin IC which is capable of controlling two DC motor simultaneously in any direction. This driver works based on H-bridge concept. In an H-bridge circuit, it allows voltage to flow in one direction. The change in voltage flow direction will make the motor to rotate either in clockwise or anti clockwise direction. In this motor driver there are two H-bridge circuit which can rotate two dc motors independently. Figure 32 illustrate the pin diagram of L293D motor controller.

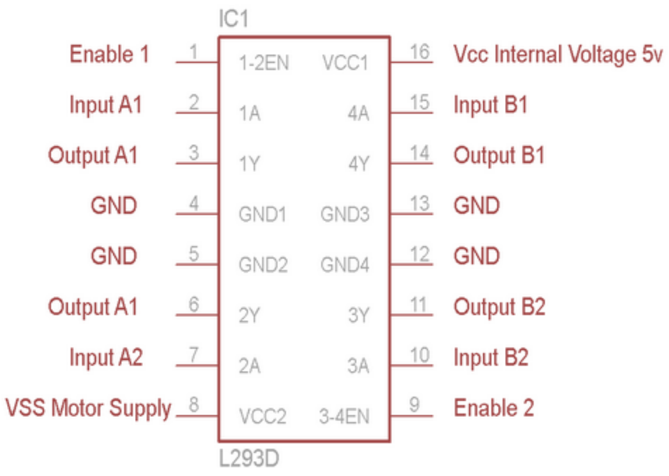


Figure 32 L293D Pin Diagram

The enable 1 and enable 2 pin should be high in order to drive the motor. In our design there are simply connected to +5V. The other inputs that need to be considered are pin 2, 7 and pin 15, 10. Both this pins are responsible to control the direction of the motor. The signals that need to be given are logic 0 and 1. Below is the table of logic combination and the rotation of the motor.

Pin 2 = Logic 1 and Pin 7 = Logic 0	**Clockwise Direction**
Pin 2 = Logic 0 and Pin 7 = Logic 1	**Anticlockwise Direction**
Pin 2 = Logic 0 and Pin 7 = Logic 0	**Idle[No rotation] Hi Impedence state**
Pin 2 = Logic 1 and Pin 7 = Logic 1	**Idle [No rotation]**

Table 1 Motor Rotation

This logic applies to the other motor as well. Figure below shows the circuit diagram of the L293D motor driver IC controller.

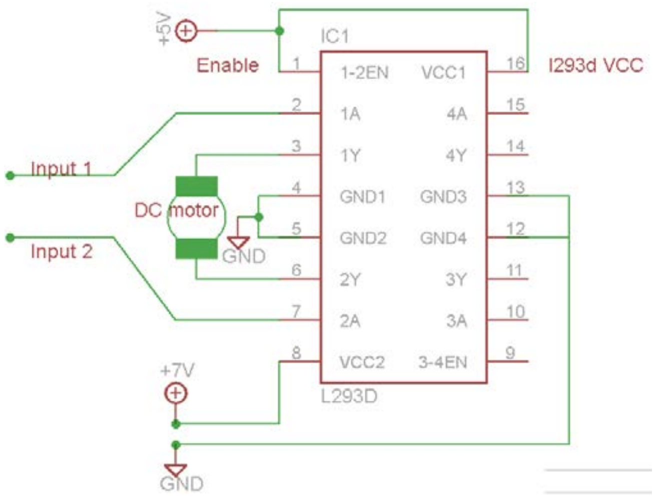


Figure 33 L293D Circuit Diagram

Figure below illustrate the actual circuit board used in building the robot.

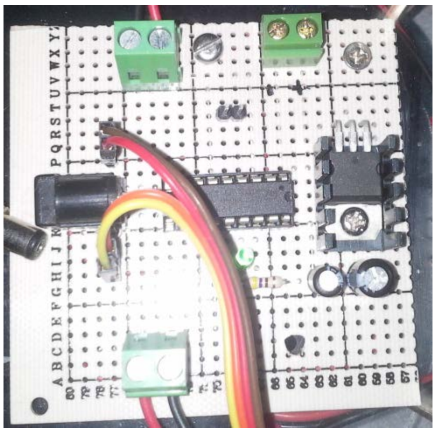


Figure 34 L293D

CHAPTER 5

CONCLUSION

In building this robot, there are a lot of lesson learned and challenges faced. Some of the challenges have been overcome and some are still in progress. The calibration of each the module was successful. The investigation begins from the Raspberry Pi, where the basic handling of Raspberry Pi is learned and progressively obtains skill and knowledge to program the Raspberry Pi to do specific function. And the next module that is being investigated is the CMUcam2.

Similar to Raspberry Pi, the CMUcam2 is investigated starting from the basic configuration using the GUI software. Later move on to control the camera via serial communication. The author faces challenges in making the Raspberry Pi to communicate with CMUcam2. The communication between these two devices is done through serial communication. The author face difficulties in capturing the data send from the CMUCam2 using the Raspberry Pi. The program that has been coded could not capture the data from the CMUCam2.

Moreover there are a lot of lesson throughout building this project. The authors were able to apply C programming language. Also the author was able to learn Linux operating system. In building the hardware of the robot, electronic and communication knowledge were applied to build motor controller circuit and also to establish the serial communication between the Raspberry Pi and the CMUCam2 the serial protocol is employed.

CHAPTER 6

RECOMMENDATIONS

In future more functionality can be added to this robot. Rather than just to follow one color, the robot can be programmed to follow green and blue color as well. The other function that can be added to this is to recognize the shape of the object as well. Apart from that, the capability of the Raspberry Pi allows more function to be added to the robot. Some of the easily addition to the robot that can be implemented is to transfer the data received by the robot via Wi-Fi. This robot can be applied in various fields. For instance it can used as firefighting robot and the imaging system can be employed in the following the traffic light as well.

REFERENCES

[1] *L. Mingliang and W. Xinqiang, "The design of intelligent robot based on embedded system," in Advanced Mechatronic Systems (ICAMechS), 2011 International Conference on, 2011, pp. 23-28.*

[2] *H. Xia and N. Houshangi, "A vision-based autonomous lane following system for a mobile robot," in Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on, 2009, pp. 2344-2349.*

[3] *M. M. Shaikh, B. Wook, L. Changhun, K. Kwang-soo, L. Tae-jae, K. Kwang-soo, et al., "Mobile robot vision tracking system using Unscented Kalman Filter," in System Integration (SII), 2011 IEEE/SICE International Symposium on, 2011, pp. 1214-1219.*

[4] *A. Rowe, C. Rosenberg, and I. Nourbakhsh, "A Second Generation Low Cost Embedded Color Vision System," in Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on, 2005, pp. 136-136.*

[5] *W. Kai and Y. Junzhi, "An embedded vision system for robotic fish navigation," in Computer Application and System Modeling (ICCASM), 2010 International Conference on, 2010, pp. V4-333-V4-337.*

[6] *B. Browning and M. Veloso, "Real-time, adaptive color-based robot vision," in Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on, 2005, pp. 3871-3876.*

[7] *M. Manigandan, G. Malathi, and N. Madhavi, "Wireless vision based moving object tracking robot through perceptual color space," in Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on, 2010, pp. 20-25.*

[8] *G. S. A. Kumar, R. Mercado, G. Manimaran, and D. T. Rover, "Enhancing student learning with hands-on RTOS development in real-time systems course," in Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, 2008, pp. S2H-11-S2H-16.*

[9] *Y. Qiang, W. Hongxing, L. Miao, and W. Tianmiao, "A novel multi-OS architecture for robot application," in Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on, 2011, pp. 2301-2306.*

[10] *T. Su-Lim and A. Tran Nguyen Bao, "Real-time operating system (RTOS) for small (16-bit) microcontroller," in Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on, 2009, pp. 1007-1011.*

[11] *B. Chun-yue, L. Yun-peng, and W. Ren-fang, "Research of key technologies for embedded Linux based on ARM," in Computer Application and System Modeling (ICCASM), 2010 International Conference on, 2010, pp. V8-373-V8-378.*

[12] *A. Gopalakrishnan, S. Greene, and A. Sekmen, "Vision-based mobile robot learning and navigation," in Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on, 2005, pp. 48-53.*

[13] *J. Palacin, A. Sanuy, X. Clua, G. Chapinal, S. Bota, M. Moreno, et al., "Autonomous mobile mini-robot with embedded CMOS vision system," in IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the], 2002, pp. 2439-2444 vol.3.*

[14] *M. Humphries, P. Radev, and M. Shirvaikar, "A realtime vehicle tracking system," in System Theory, 2005. SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on, 2005, pp. 357-361.*

Appendices

Appendix A. Serial Code

Headers required

```
#include <stdio.h>
#include <unistd.h>                              //Used for UART
#include <fcntl.h>                               //Used for UART
#include <termios.h>                             //Used for UART
```

Setting Up The UART

```
        //-------------------------
        //----- SETUP USART 0 -----
        //-------------------------
        //At bootup, pins 8 and 10 are already set to UART0_TXD, UART0_RXD (ie the
        //alt0 function) respectively
        int uart0_filestream = -1;

        //OPEN THE UART
        //The flags (defined in fcntl.h):
        //       Access modes (use 1 of these):
        //               O_RDONLY - Open for reading only.
        //               O_RDWR - Open for reading and writing.
        //               O_WRONLY - Open for writing only.
        //
        //       O_NDELAY / O_NONBLOCK (same function) - Enables nonblocking mode.
        //When set read requests on the file can return immediately with a failure
        //status
        //
        //
        //
        //       O_NOCTTY - When set and path identifies a terminal device, open()
        //shall not cause the terminal device to become the controlling terminal
        //for the process.
        uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);
        //Open in non blocking read/write mode
        if (uart0_filestream == -1)
        {
                //ERROR - CAN'T OPEN SERIAL PORT
             printf("Error - Unable to open UART.  Ensure it is not in use by
             another application\n");
        }

        //CONFIGURE THE UART
        //The flags (defined in /usr/include/termios.h)
        //Baud rate:- B1200, B2400, B4800, B9600, B19200, B38400, B57600, B115200,
        //B230400, B460800, B500000, B576000, B921600, B1000000, B1152000,
        //B1500000, B2000000, B2500000, B3000000, B3500000, B4000000
        //       CSIZE:- CS5, CS6, CS7, CS8
        //       CLOCAL - Ignore modem status lines
        //       CREAD - Enable receiver
        //       IGNPAR = Ignore characters with parity errors
        //       ICRNL - Map CR to NL on input (Use for ASCII comms where you want
        //        to auto correct end of line characters - don't use for bianry //
        //        comms!)
        //       PARENB - Parity enable
        //       PARODD - Odd parity (else even)
        struct termios options;
        tcgetattr(uart0_filestream, &options);
        options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;          //<Set baud rate
        options.c_iflag = IGNPAR;
        options.c_oflag = 0;
        options.c_lflag = 0;
        tcflush(uart0_filestream, TCIFLUSH);
        tcsetattr(uart0_filestream, TCSANOW, &options);
```

Transmitting Bytes

```
        //----- TX BYTES -----
        unsigned char tx_buffer[20];
```

```
	unsigned char *p_tx_buffer;

	p_tx_buffer = &tx_buffer[0];
	*p_tx_buffer++ = 'H';
	*p_tx_buffer++ = 'e';
	*p_tx_buffer++ = 'l';
	*p_tx_buffer++ = 'l';
	*p_tx_buffer++ = 'o';

	if (uart0_filestream != -1)
	{
		int count = write(uart0_filestream, &tx_buffer[0], (p_tx_buffer -
		&tx_buffer[0]));
		//Filestream, bytes to write, number of bytes to write
		if (count < 0)
		{
			printf("UART TX error\n");
		}
	}
```

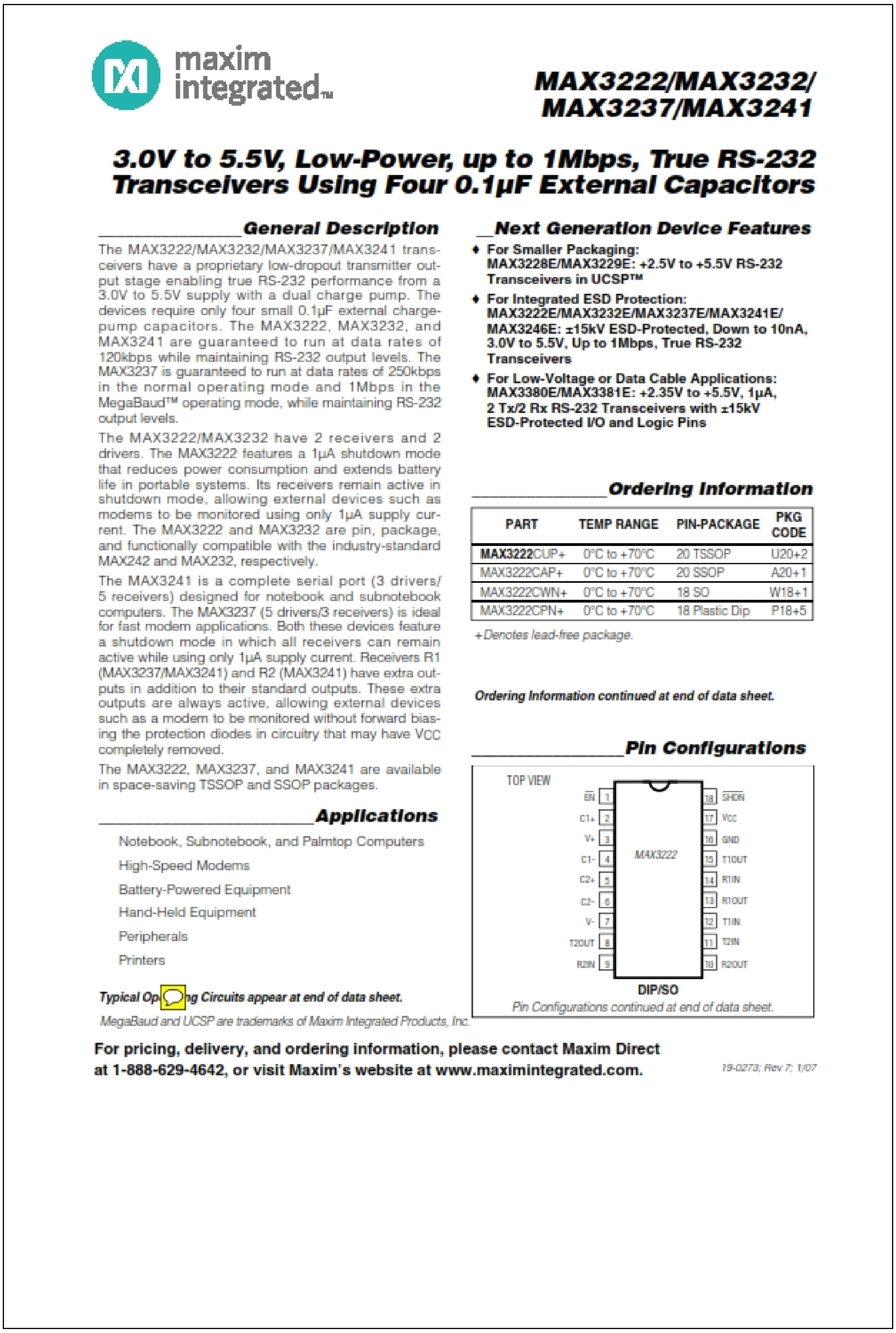
Receiving Bytes

```
	//----- CHECK FOR ANY RX BYTES -----
	if (uart0_filestream != -1)
	{
		// Read up to 255 characters from the port if they are there
		unsigned char rx_buffer[256];
		int rx_length = read(uart0_filestream, (void*)rx_buffer, 255);
		if (rx_length < 0)
		{
			//An error occured (will occur if there are no bytes)
		}
		else if (rx_length == 0)
		{
			//No data waiting
		}
		else
		{
			//Bytes received
			rx_buffer[rx_length] = '\0';
			printf("%i bytes read : %s\n", rx_length, rx_buffer);
		}
	}
```

Closing the UART if no longer needed

```
	//----- CLOSE THE UART -----
	close(uart0_filestream);
```

Appendix B. MAX3232 Datasheet



maxim integrated

MAX3222/MAX3232/ MAX3237/MAX3241

3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1µF External Capacitors

General Description

The MAX3222/MAX3232/MAX3237/MAX3241 transceivers have a proprietary low-dropout transmitter output stage enabling true RS-232 performance from a 3.0V to 5.5V supply with a dual charge pump. The devices require only four small 0.1µF external charge-pump capacitors. The MAX3222, MAX3232, and MAX3241 are guaranteed to run at data rates of 120kbps while maintaining RS-232 output levels. The MAX3237 is guaranteed to run at data rates of 250kbps in the normal operating mode and 1Mbps in the MegaBaud™ operating mode, while maintaining RS-232 output levels.

The MAX3222/MAX3232 have 2 receivers and 2 drivers. The MAX3222 features a 1µA shutdown mode that reduces power consumption and extends battery life in portable systems. Its receivers remain active in shutdown mode, allowing external devices such as modems to be monitored using only 1µA supply current. The MAX3222 and MAX3232 are pin, package, and functionally compatible with the industry-standard MAX242 and MAX232, respectively.

The MAX3241 is a complete serial port (3 drivers/ 5 receivers) designed for notebook and subnotebook computers. The MAX3237 (5 drivers/3 receivers) is ideal for fast modem applications. Both these devices feature a shutdown mode in which all receivers can remain active while using only 1µA supply current. Receivers R1 (MAX3237/MAX3241) and R2 (MAX3241) have extra outputs in addition to their standard outputs. These extra outputs are always active, allowing external devices such as a modem to be monitored without forward biasing the protection diodes in circuitry that may have V_{CC} completely removed.

The MAX3222, MAX3237, and MAX3241 are available in space-saving TSSOP and SSOP packages.

Applications

Notebook, Subnotebook, and Palmtop Computers
High-Speed Modems
Battery-Powered Equipment
Hand-Held Equipment
Peripherals
Printers

Typical Op[illegible]g Circuits appear at end of data sheet.

MegaBaud and UCSP are trademarks of Maxim Integrated Products, Inc.

Next Generation Device Features

- **For Smaller Packaging: MAX3228E/MAX3229E: +2.5V to +5.5V RS-232 Transceivers in UCSP™**
- **For Integrated ESD Protection: MAX3222E/MAX3232E/MAX3237E/MAX3241E/ MAX3246E: ±15kV ESD-Protected, Down to 10nA, 3.0V to 5.5V, Up to 1Mbps, True RS-232 Transceivers**
- **For Low-Voltage or Data Cable Applications: MAX3380E/MAX3381E: +2.35V to +5.5V, 1µA, 2 Tx/2 Rx RS-232 Transceivers with ±15kV ESD-Protected I/O and Logic Pins**

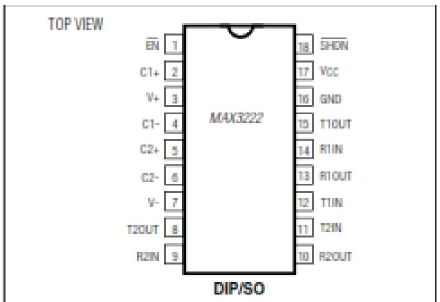
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	PKG CODE
MAX3222CUP+	0°C to +70°C	20 TSSOP	U20+2
MAX3222CAP+	0°C to +70°C	20 SSOP	A20+1
MAX3222CWN+	0°C to +70°C	18 SO	W18+1
MAX3222CPN+	0°C to +70°C	18 Plastic Dip	P18+5

+Denotes lead-free package.

Ordering Information continued at end of data sheet.

Pin Configurations



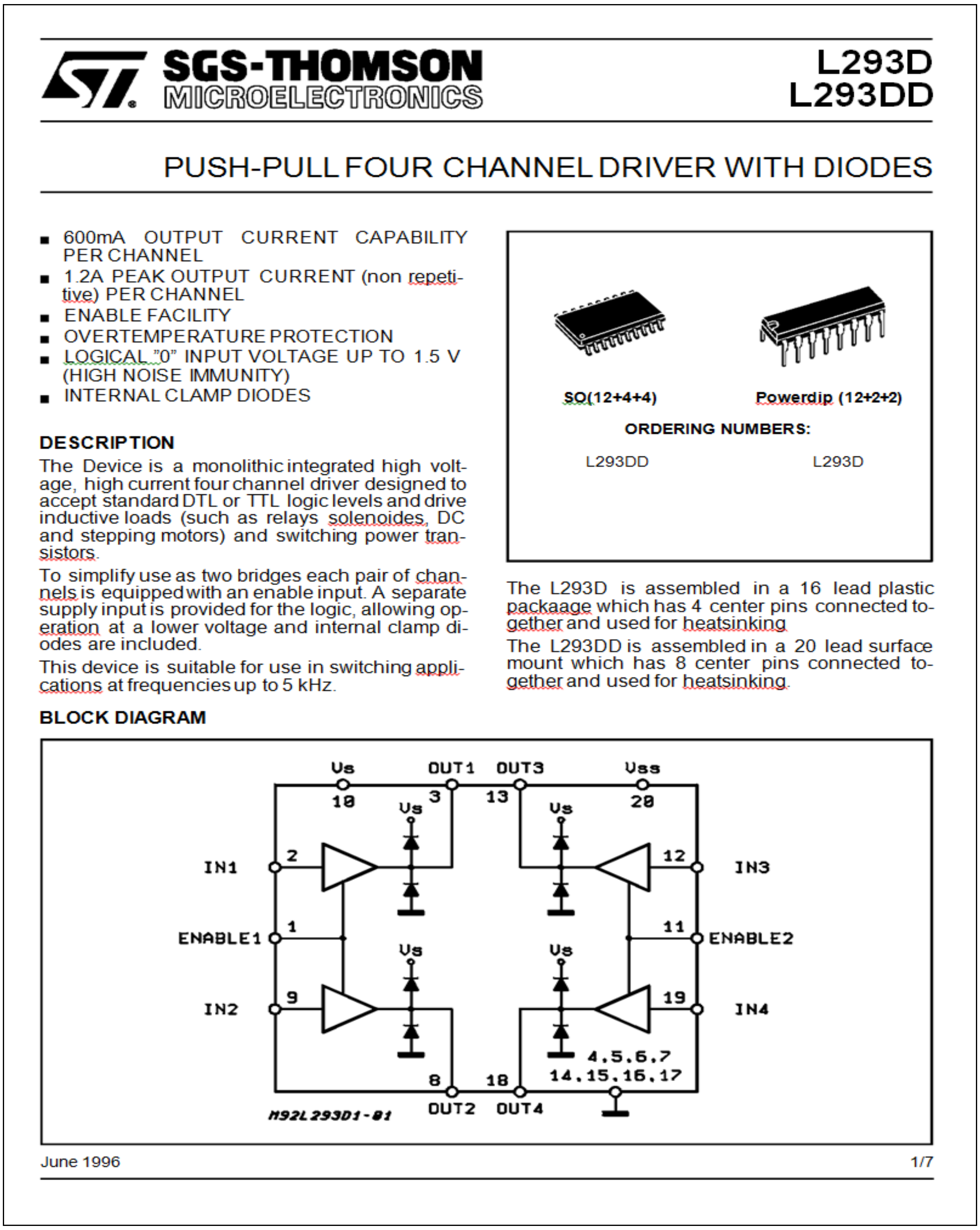
DIP/SO

Pin Configurations continued at end of data sheet.

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.

19-0273; Rev 7; 1/07

Appendix C. L293D Datasheet



SGS-THOMSON MICROELECTRONICS

L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES

SO(12+4+4) **Powerdip (12+2+2)**

ORDERING NUMBERS:

L293DD	L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

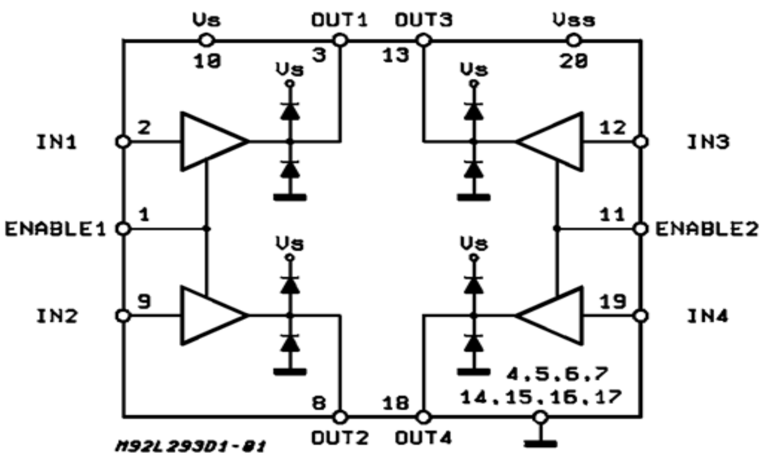
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking.

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

BLOCK DIAGRAM



June 1996 1/7