CHAPTER: 1

INTRODUCTION

1.1 OVERVIEW

According to the present scenario, it is difficult to carry heavy luggage around manually. In India there is good security at the Airports, but at Train stations and Bus stands there are a lot of cases of robbery and damage of luggage. Hence taking all these points under consideration we have implemented a Smart Luggage Carrier.

1.2 INTRODUCTION

Most of the people around the world travel through trains and airplanes, for longer journeys. These people usually have heavy luggage's with them. In Railways stations today the luggage's are carried by coolies and at higher rates. Sometimes these luggage's are misplaced or stolen. So to overcome these problems a smart and sophisticated method has to be found out which will help the travelers and also reduce the number of robbery at such places.

1.3 PROBLEM STATEMENT

Now a day's carrying luggage's is done manually or using trolleys at Bus stations, Railways station and Airports. Hence in this project we develop a SMART way of carrying luggage.

1.4 OBJECTIVES

The primary objective is to build a system which provides an easy way for the transportation or carrying the luggage.

1.5 ORGANIZATION OF THE REPORT

The report includes complete structure of the Smart Luggage Carrier system as follows.

Literature survey consists of a paper published by different authors on the related project, along with existing systems, their advantages and drawbacks. The survey sheds light on the idea of proposed model and its advantages over previous systems. Next comes the requirement analysis and specifications of the proposed system. Requirement analysis explains about functional and non-functional aspects of the project and also hardware and software requirements along with their specifications. After analyzing requirements one can start with the design process. Our system design includes complete description about the architecture and the module. Implementation is done using all the data collected and by configuring hardware and software modules as per our needs. Implemented system can be tested for the assurance of its working under different sets of input parameters to get the expected results. Next chapter includes the result of the system as whole and system analysis with different cases of all the applications.

Conclusion and future work enlightens many more applications and advantages of the proposed system and also gives idea about further development.

CHAPTER: 2

LITERATURE SURVEY

2.1 OVERVIEW

In the research paper titled Automated Luggage Carrying System by Md. Imran Khan, Saad Bin Siddique, Nazmul Hassan, Md.Towhid Chowdhury[1] they mentioned about an Integument personal luggage carrying system can add comfort to the explorer inside an infrastructure possibly airport. The system consists of automated vehicles that can be borrowed and it automatically follows the borrower inside an infrastructure with luggage. These also maintain a distance from the borrower and do not follow in some of the restricted areas like restroom for comfort. After service these will automatically move to the docking station for charging and reuse. In this thesis we have identified the basic construction required for the six wheels based rigid robot body and the basic person tracking movement of the smart card holder for airport traveler's personal luggage carrier

2.2 EXISTING SYSTEM

In the present time, there are luggage carriers or trolleys which have to move around manually. Due to this a lot of physical strength is required and a watch must always be kept on the luggage so that it is not stolen. In case the luggage is heavier then it's harder to move around with the cart. As proposed in the Literature paper titled Automated Luggage Carrying System [1] they have a used Six wheel technology which is used for locomotion, and sensor for tracking the cart.

Disadvantages:

- A lot of power is consumed by using Six wheel Technology.
- Placing sensors for tracking the cart in the whole area would increase the budget tremendously.
- Security of the luggage is most important which is not provided.

2.3 PROPOSED SYSTEM

In this project we present a smarter way to move around with luggage or any items may it be Shopping Mall, Railways station, Airport or Hospital. In the future when there will more sophistication a need for smarter ways to carry luggage and have security for it will arise, which is the primary purpose of our project. The proposed Smart Luggage Carrier is a small prototype. Here we are using a “Three Wheel Technology” as used in Auto rickshaw and Image processing for the Locomotion. Also adding security features for the luggage in the cart and techniques to avoid object collusion.

CHAPTER: 3

REQUIREMENT ANALYSIS AND SPECIFICATION

3.1 OVERVIEW

The process of studying and analyzing the customer and the user needs to arrive at a definition of the problem domain and system requirements. It includes discovering the boundaries of the new system or software and how it must interact with its environment within the new problem domain and detecting and resolving conflicts between user requirements.

3.2 FUNCTIONAL REQUIREMENTS

The functional requirements consist of the following input and output values:

3.2.1 Positive Images

Positive samples are created by opencv_createsamples utility. They may be created from a single image with object or from a collection of previously marked up images.Positive images are the images which the cart detects as a pattern which will be strapped onto the leg of the user which helps the cart to locate and follow the user.

The positive images are detected by the technology called OpenCV[2], which is used in image processing to detect the pattern which helps the cart to follow the user seen by the camera which is a part of the cart. The camera continuously tracks the user to follow him

3.2.2 Ultrasonic sensor

The ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1” to 13 feet. It’s operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It is used in the cart so that it doesn’t collide with any objects on it way while following the user.

3.2.3 PIR sensors for alarm

Passive Infrared sensors are used to detect whether a human has moved in or out of the sensors range. It has Sensitivity range**:** up to 20 feet (6 meters) 110° x 70° detection range. Power supply**:** 3V-9V input voltage, but 5V is ideal. The PIR sensor is used as an intrusion detection in the cart so the luggage is secured while it follows the user.PIR sensor along with piezo buzzer is attached to the cart so that whenever the intrusion is detected while following the user the alarm beeps.

3.2.4 PAN movement

When the pattern is recognised by the camera attached to the PAN which is in turn attached to the cart PAN movement takes place, it moves in such a way that it tracks the pattern strapped to the user .The servo motor of the pan has 180 Degree rotation

3.2.5 Movement of wheels

When the pattern which is strapped to the user is recognised, the DC Motors of the cart move at a certain speed until the pattern is still in sight of the camera. Once the pattern is not visible the cart stops at the same time.

3.2.6 Direction Wheel

The movement of the direction wheel is same as that of the Pan. As both the servo's of the Pan and the direction wheel take the same inputs ,they move at the same time in the same direction.

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 Performance

Performance is the capabilities of a machine or vehicle. The hardware is put through tests which assess the performance of the processor Performance measurement is the process of collecting, analysing and reporting information regarding the performance of an component.

The performance of the cart depends on the components of the cart and the technologies used in the cart. Opencv is a growing technology and is a effective way of approaching image processing technique. So in the software the performance is very high. Proper maintenance of the cart will increases the performance of the cart.

3.3.2 Maintenance

Maintenance, repair and operations or maintenance, repair, and overhaul involve fixing any sort of mechanical, plumbing or electrical device should it become out of order or broken (known as repair, unscheduled, or casualty maintenance). It also includes performing routine actions which keep the device in working order. Maintenance may be defined as, "All actions which have the objective of retaining or restoring an item in or to a state in which it can perform its required function. The actions include the combination of all technical and corresponding administrative, managerial, and supervision actions."

Maintenance operations can be categorised by whether the product remains the property of the customer, i.e. a service is being offered, or whether the product is bought by the reprocessing organisation and sold to any customer wishing to make the purchase .In the former case it may be a backshop operation within a larger organization or smaller operation.

The former of these represents a closed loop supply chain and usually has the scope of maintenance, repair or overhaul of the product. The latter of the categorisations is an open loop supply chain and is typified by refurbishment and remanufacture. The main characteristic of the closed loop system is that the demand for a product is matched with the supply of a used product. Neglecting asset write-offs and exceptional activities the total population of the product between the customer and the service provider remains constant.

For the maintenance of the cart the circuits like raspberry and Arduino are covered by the box for the outside moisture of water and dust so the maintenance of the cart is very low. The user should only take care that he should not exceed the threshold value of power supply otherwise the circuits may get damaged

3.3.3 Capacity

After testing the we have come to the conclusion that the capacity of the cart is 3.7kg.The capacity of the cart can be increased by increasing the power given to the dc motor which in turn accelerates the cart. The power can be 12V at the maximum

3.3.4 Security

Security here refers to the security of the luggage and the Components of the cart. This Smart Luggage carrier is equipped with motion sensors in the carrier section of the cart for security. The other components of The Smart Luggage Carrier packed under the cart such that only the technician can open it or configure it.

3.4 Hardware Requirements

3.4.1 Raspberry Pi

3.4.1.1 Hardware Layout

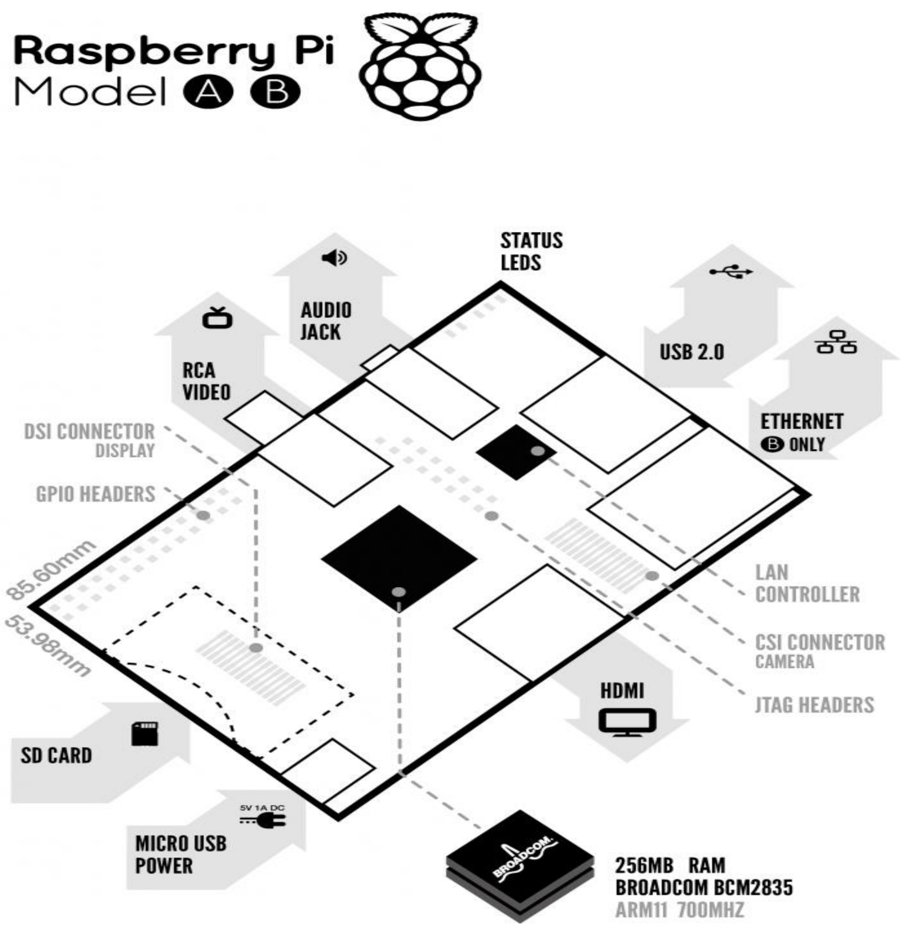


Fig 3.1: Block Diagram of Raspberry Pi

3.4.1.2 A brief description of the components on the Pi.

1) Processor / SoC (System on Chip)

The Raspberry Pi has a Broadcom BCM2835 System on Chip module. It has a ARM1176JZF-S processor

The Broadcom SoC used in the Raspberry Pi[3] is equivalent to a chip used in an old smartphone (Android or iPhone). While operating at 700 MHz by default, the Raspberry Pi provides a real world performance roughly equivalent to the 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997-1999, but the GPU, however, provides 1 Gpixel/s, 1.5 Gtexel/s or 24 GFLOPS of general purpose compute and the graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox of 2001. The Raspberry Pi chip operating at 700 MHz by default, will not become hot enough to need a heat sink or special cooling.

2) Power source

The Pi is a device which consumes 700mA or 3W or power. It is powered by a MicroUSB charger or the GPIO header. Any good smart phone charger will do the work of powering the Pi.

3) SD Card

The Raspberry Pi does not have any onboard storage available. The operating system is loaded on a SD card which is inserted on the SD card slot on the Raspberry Pi. The operating system can be loaded on the card using a card reader on any computer.

4) GPIO– General Purpose Input Output

General-purpose input/output (GPIO) is a generic pin on an integrated circuit whose behaviour, including whether it is an input or output pin, can be controlled by the user at run time.

GPIO pins have no special purpose defined, and go unused by default. The idea is that sometimes the system designer building a full system that uses the chip might find it useful to have a handful of additional digital control lines, and having these available from the chip can save the hassle of having to arrange additional circuitry to provide them.

GPIO capabilities may include:

- GPIO pins can be configured to be input or output
- GPIO pins can be enabled/disabled
- Input values are readable (typically high=1, low=0)
- Output values are writable/readable
- Input values can often be used as IRQs (typically for wakeup events)

The production Raspberry Pi board has a 26-pin 2.54 mm (100 mil) expansion header, marked as P1, arranged in a 2x13 strip. They provide 8 GPIO pins plus access to I²C, SPI, UART), as well as +3.3 V, +5 V and GND supply lines. Pin one is the pin in the first column and on the bottom row.

Fig 3.2: GPIO connector on RPi

5) DSI Connector

The Display Serial Interface (DSI) is a specification by the Mobile Industry Processor Interface (MIPI) Alliance aimed at reducing the cost of display controllers in a mobile device. It is commonly targeted at LCD and similar display technologies. It defines a serial bus and a communication protocol between the host (source of the image data) and the device (destination of the image data).

A DSI compatible LCD screen can be connected through the DSI connector, although it may require additional drivers to drive the display.

6) RCA Video

RCA Video outputs (PAL and NTSC) are available on all models of Raspberry Pi. Any television or screen with a RCA jack can be connected with the RPi.

7) Audio Jack

A standard 3.5 mm TRS connector is available on the RPi for stereo audio output. Any headphone or 3.5mm audio cable can be connected directly. Although this jack cannot be used for taking audio input, USB mics or USB sound cards can be used.

8) Status LEDs

There are 5 status LEDs on the RPi that show the status of various activities as follows:

“OK” - SDCard Access (via GPIO16) - labeled as "OK" on Model B Rev1.0 boards and "ACT" on Model B Rev2.0 and Model A boards

“POWER” - 3.3 V Power - labeled as "PWR" on all boards.

“FDX” - Full Duplex (LAN) **(Model B)** - labelled as "FDX" on all boards.

“LNK” - Link/Activity (LAN) **(Model B)** - labelled as "LNK" on all boards.

9) USB 2.0 Port

USB 2.0 ports are the means to connect accessories such as mouse or keyboard to the Raspberry Pi. There is 1 port on Model A, 2 on Model B and 4 on Model B+. The number of ports can be increased by using an external powered USB hub which is available as a standard Pi accessory.

10) Ethernet

Ethernet port is available on Model B and B+. It can be connected to a network or internet using a standard LAN cable on the Ethernet port. The Ethernet ports are controlled by Microchip LAN9512 LAN controller chip.

11) CSI connector

CSI – Camera Serial Interface is a serial interface designed by MIPI (Mobile Industry Processor Interface) alliance aimed at interfacing digital cameras with a mobile processor.

The RPi foundation provides a camera specially made for the Pi which can be connected with the Pi using the CSI connector.

12) JTAG headers

JTAG is an acronym for ‘Joint Test Action Group', an organization that started back in the mid 1980's to address test point access issues on PCB with surface mount devices. The organization devised a method of access to device pins via a serial port that became known as the TAP (Test Access Port). In 1990 the method became a recognized international standard (IEEE Std 1149.1). Many thousands of devices now include this standardized port as a feature to allow test and design engineers to access pins.

13) HDMI

HDMI – High Definition Multimedia Interface

HDMI 1.3 a type A port is provided on the RPi to connect with HDMI screens.

3.4.2 Arduino Uno

3.4.2.1Hardware Layout

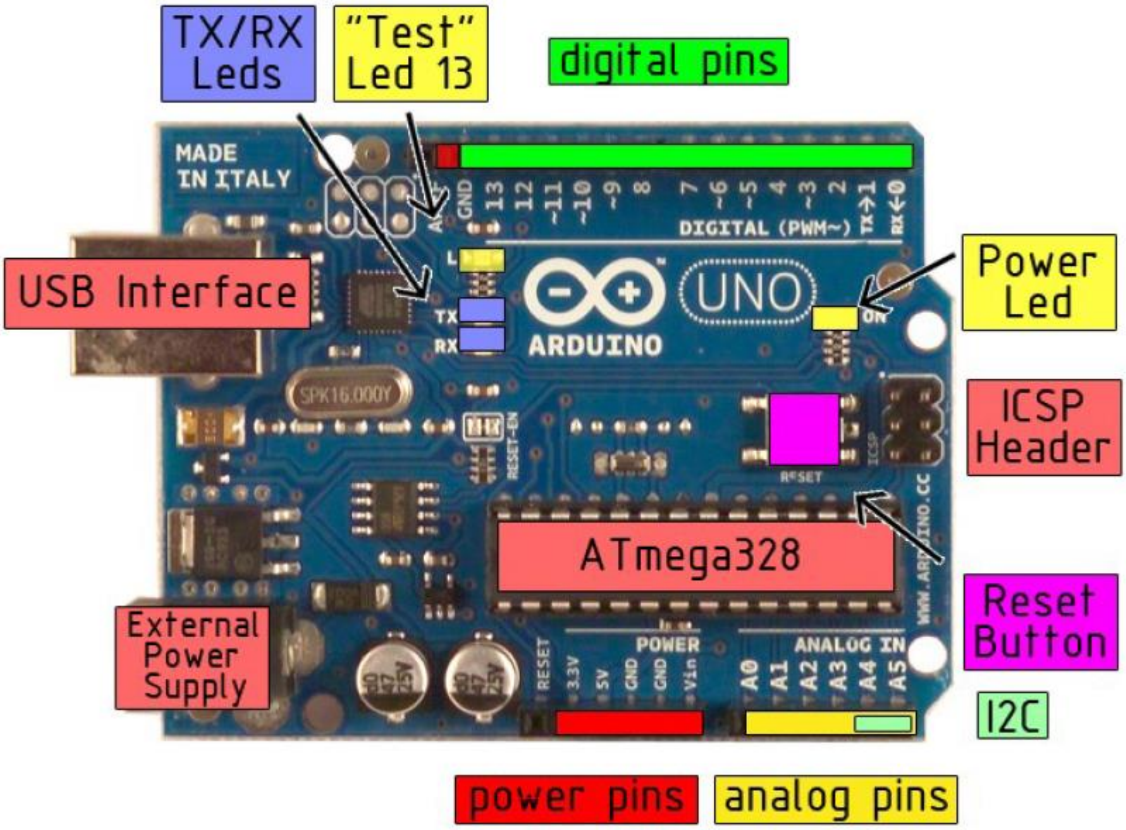


Fig 3.3: Arduino Board

The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0[4]. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of board

3.4.2.2 Components of Uno Board:

Microcontroller	**ATmega328P**
Operating Voltage	**5V**
Input Voltage (recommended)	**7-12V**
Input Voltage (limit)	**6-20V**
Digital I/O Pins	**14 (of which 6 provide PWM output)**
PWM Digital I/O Pins	**6**
Analog Input Pins	**6**
DC Current per I/O Pin	**20 mA**
DC Current for 3.3V Pin	**50 mA**
Flash Memory	**32 KB (ATmega328P) of which 0.5 KB used by bootloader**
SRAM	**2 KB (ATmega328P)**
EEPROM	**1 KB (ATmega328P)**
Clock Speed	**16 MHz**
Length	**68.6 mm**
Width	**53.4 mm**
Weight	**25 g**

Table 3.1:Components of Uno Board

3.4.2.3 Brief description of the components on the Uno Board:

1) Power:

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:

• 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.

• 5V : The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

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

. • GND : Ground pins

- IOREF: This pin on the Uno 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 to work with the 5V or 3.3V.

2) Memory :

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM.

3) Input and Output :

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

• Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .

• External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

• PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

• SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

• LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

• I 2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library. There are a couple of other pins on the board:

• AREF. Reference voltage for the analog inputs. Used with analogReference().

• Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-toserial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

4) Automatic (Software) Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN"

3.5 Software Requirements

3.5.1 Operating System of Raspberry Pi

The Raspberry Pi primarily uses Linux kernel-based operating systems. The ARM11 is based on version 6 of the ARM which is no longer supported by several popular versions of Linux, including Ubuntu. The install manager for Raspberry Pi is NOOBS.

The OSs included with NOOBS are:

- Archlinux ARM
- OpenELEC
- Pidora (Fedora Remix)
- Raspbmc and the XBMC open source digital media center
- RISC OS – The operating system of the first ARM-based computer
- Raspbian (recommended) – Maintained independently of the Foundation; based on ARM hard-float (armhf)-Debian 7 'Wheezy' architecture port, that was designed for a newer ARMv7 processor whose binaries would not work on the Rapberry Pi, but Raspbian is compiled for the ARMv6 instruction set of the Raspberry Pi making it work but with slower performance. It provides some available deb software packages, pre-compiled software bundles. A minimum size of 2 GB SD card is required, but a 4 GB SD card or above is recommended. There is a Pi Store for exchanging programs. The 'Raspbian Server Edition (RSEv2.4)', is a stripped version with other software packages bundled as compared to the usual desktop computer oriented Raspbian.

3.3.1 Boot Process

The Raspberry Pi does not boot as a traditional computer. The Video Core i.e. the Graphics processor actually boots before the ARM CPU.

The boot process of the Raspberry Pi can be explained as follows:

- When the power is turned on, the first bits of code to run is stored in a ROM chip in the SoC and is built into the Pi during manufacture. This is the called the **first-stage bootloader**.
- The SoC is hardwired to run this code on startup on a small RISC Core (**R**educed **I**nstruction **S**et **C**omputer). It is used to mount the FAT32 boot partition in the SDCard so that the **second-stage bootloader** can be accessed. So what is this 'second-stage bootloader' stored in the SD Card? It's **'bootcode.bin'.** This file can be seen while mount process of an operating system on the SD Card in windows.
- Now here's something tricky. The first-stage bootloader has not yet initialized the ARM CPU (meaning CPU is in **reset**) or the RAM. So, the second-stage bootloader also has to run on the GPU. The bootloader.bin file is loaded into the 128K 4 way set associative L2 cache of the GPU and then executed. This enables the RAM and loads **start.elf** which is also in the SD Card. This is the **third-stage bootloader** and is also the most important. It is the firmware for the GPU, meaning it contains the settings or in our case, has instructions to load the settings from **config.txt** which is also in the SD Card. We can think of the config.txt as the 'BIOS settings'.
- The **start.elf** also splits the RAM between the GPU and the ARM CPU. The ARM only has access the to the address space left over by the GPU address space. For example, if the GPU was allocated addresses from 0x000F000 – 0x0000FFFF, the ARM has access to addresses from 0x00000000 – 0x0000EFFF.
- The physical addresses perceived by the ARM core is actually mapped to another address in the VideoCore (0xC0000000 and beyond) by the MMU (Memory Management Unit) of the VideoCore.
- The config.txt is loaded **after** the split is done so the splitting amounts cannot be specified in the config.txt. However, different .elf files having different splits exist in

the SD Card. So, depending on the requirement, the file can be renamed to start.elf and boot the Pi. In the Pi, the GPU is King!

- Other than loading config.txt and splitting RAM, the **start.elf** also loads **cmdline.txt** if it exists. It contains the command line parameters for whatever kernel that is to be loaded. This brings us to the final stage of the boot process. The **start.elf** finally loads **kernel.img** which is the binary file containing the OS kernel and releases the **reset** on the CPU. The ARM CPU then executes whatever instructions in the **kernel.img** thereby loading the operating system.

3.5.2 AURDINO IDE

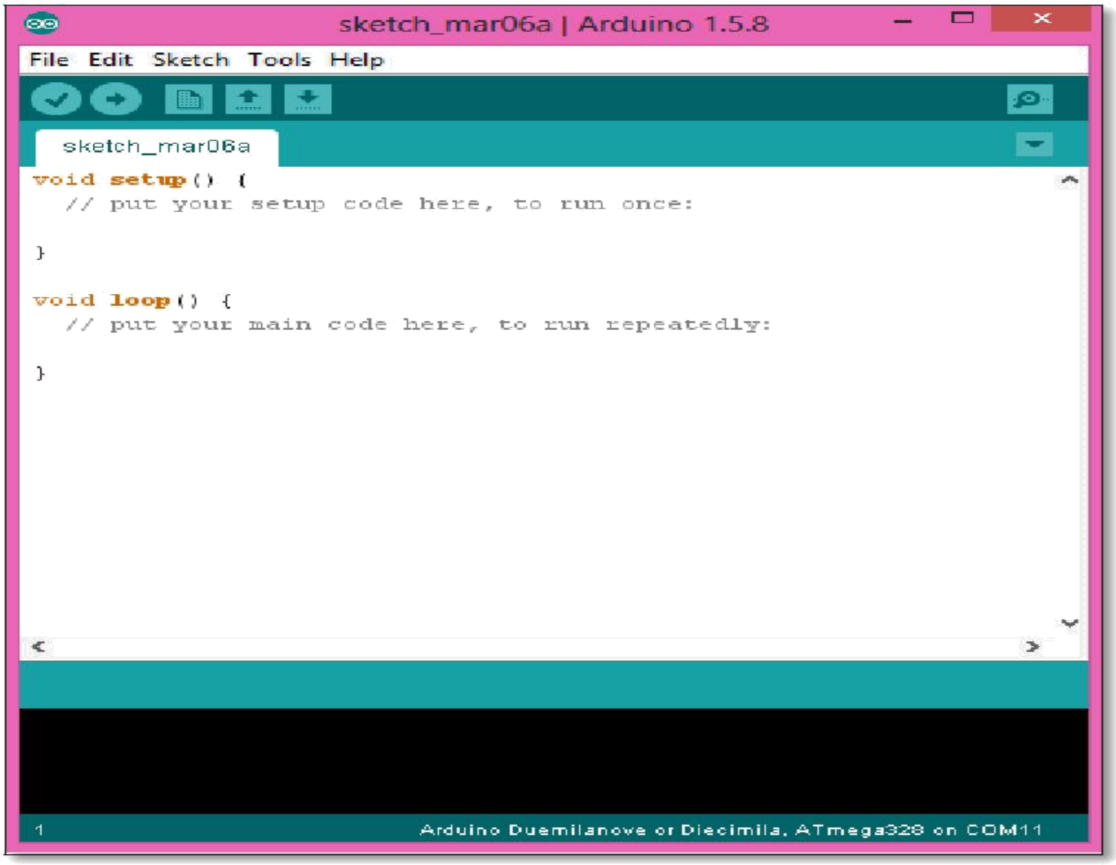


Fig 3.4:Arduino IDE

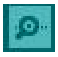
 To verify or Compile the code written in the workspace.

 To Upload the written program to Arduino Board.

 To open the new workspace or file.

To open the saved programs.

To save the written data or program.



For opening Serial Monitor (only in case of serial Communication).

Arduino programs may be written in any programming language with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in Java. It originated from the IDE for the Processing programming language project and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism for compiling and loading programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the C and C++ programming languages using special rules of code organization. The Arduino IDE supplies a software library called "Wiring" from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub main() into an executable cyclic executive program:

setup(): a function that runs once at the start of a program and that can initialize settings.

loop(): a function called repeatedly until the board powers off.

After compilation and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

CHAPTER: 4

SYSTEM DESIGN

4.1 OVERVIEW

Design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user. System design is the process of defining the architecture components, modules, interfaces, and for a system to satisfy specified requirements. Systems design could be seen as the application of system theory to product development. There is some overlap with the disciplines of system analysis, systems architecture and systems engineering.

4.2 MODULE DESCRIPTION

4.1.1 Object Detection-using Haar Cascade Classifier

A classifier[5] is trained with a few hundred sample views of a particular object (i.e., a face or an Image pattern), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (*stages*) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques.

The current algorithm uses the following Haar-like features:

1. Edge features

2. Line features

3. Center-surround features

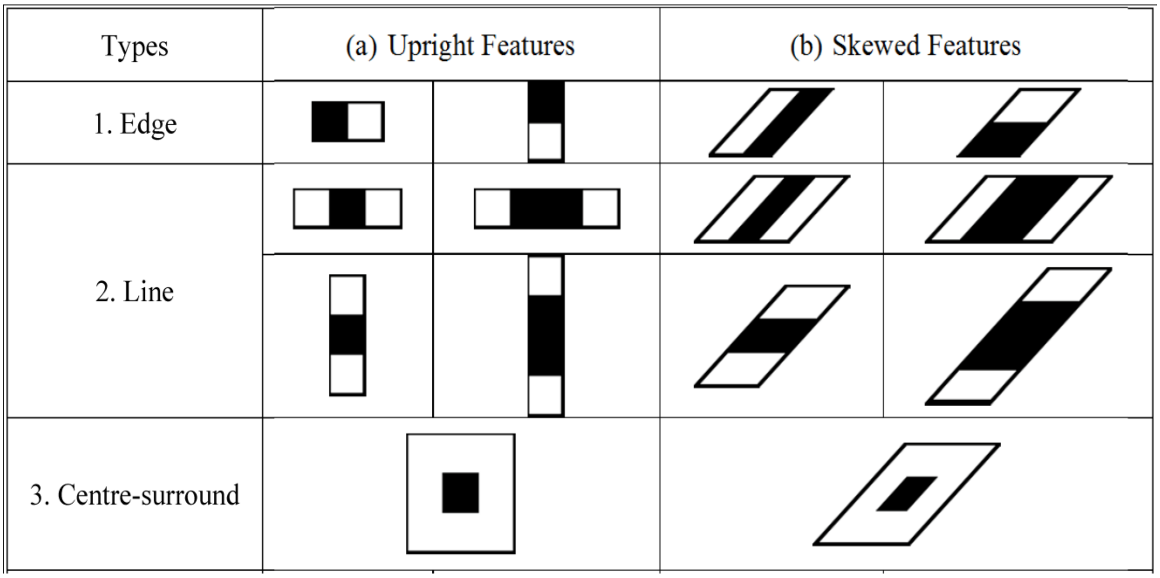


Fig 4.1: Features of Haar

The feature used in a particular classifier is specified by its shape (1a, 2b), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2b) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over rectangular regions are calculated rapidly using integral images

For training we need a set of samples. There are two types of samples: negative and positive

1.Positive Samples

Positive samples correspond to images[6] with detected objects. A small description about positive images is given in Chapter 2 Please note that you need a large dataset of positive samples before you give it to the mentioned utility, because it only applies perspective transformation. For example you may need only one positive sample for absolutely rigid object like an OpenCV logo, but you definitely need hundreds and even thousands of positive samples for faces. In the case of faces you should consider all the race and age groups, emotions and perhaps beard styles.

So, a single object image may contain a company logo. Then a large set of positive samples is created from the given object image by random rotating, changing the logo intensity as well as placing the logo on arbitrary background. The amount and range of randomness can be controlled by command line arguments of opencv_createsamples utility.

Command line arguments:

- -vec <vec_file_name>

Name of the output file containing the positive samples for training.

- -img <image_file_name>

Source object image (e.g., a company logo).

- -bg <background_file_name>

Background description file; contains a list of images which are used as a background for randomly distorted versions of the object.

- -num <number_of_samples>

Number of positive samples to generate.

- -bgcolor <background_color>

Background color (currently grayscale images are assumed); the background color denotes the transparent color. Since there might be compression artifacts, the amount of color tolerance can be specified by -bgthresh. All pixels withing bgcolor-bgthresh and bgcolor+bgthresh range are interpreted as transparent.

- -bgthresh <background_color_threshold>

- -inv

If specified, colors will be inverted.

- -randinv

If specified, colors will be inverted randomly.

- -maxidev <max_intensity_deviation>

Maximal intensity deviation of pixels in foreground samples.

- -maxxangle <max_x_rotation_angle>
- -maxyangle <max_y_rotation_angle>
- -maxzangle <max_z_rotation_angle>

Maximum rotation angles must be given in radians.

- -show

Useful debugging option. If specified, each sample will be shown. Pressing Esc will continue the samples creation process without.

- -w <sample_width>

Width (in pixels) of the output samples.

- -h <sample_height>

Height (in pixels) of the output samples.

- -pngoutput

With this option switched on opencv_createsamples tool generates a collection of PNG samples and a number of associated annotation files, instead of a single vec file.

The opencv_createsamples utility may work in a number of modes, namely:

- Creating training set from a single image and a collection of backgrounds:
  - with a single vec file as an output;
  - with a collection of JPG images and a file with annotations list as an output;
  - with a collection of PNG images and associated files with annotations as an output;

- Converting the marked-up collection of samples into a vec format;
- Showing the content of the vec file.

2. Negative Samples

Negative samples correspond to non-object images. Negative samples are taken from arbitrary images. These images must not contain detected objects. Negative samples are enumerated in a special file. It is a text file in which each line contains an image filename (relative to the directory of the description file) of negative sample image. This file must be created manually. Note that negative samples and sample images are also called background samples or background samples images, and are used interchangeably in this document. Described images may be of different sizes. But each image should be (but not necessarily) larger then a training window size, because these images are used to subsample negative image to the training size.

An example of description file:

Directory structure:

/img

img1.jpg

img2.jpg

bg.txt

File bg.txt:

img/img1.jpg

img/img2.jpg

4.1.1.1 Steps for Cascade Training

Creating a file that contains current folder file names:

dir \b > bg.txt

Creating info.lst File :

opencv_createsamples –img positive.jpeg –bg bg.txt -info info.lst -maxxangle 0.5 –maxyangle 0.5 –num 50

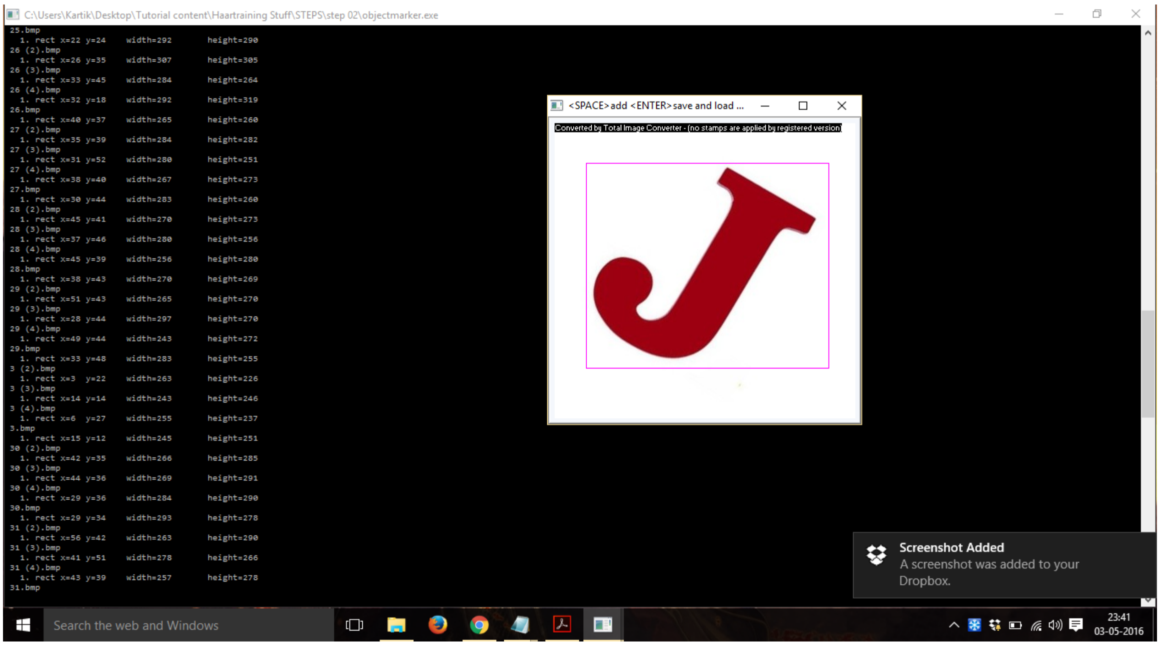


Fig 4.2:Screenshot of Cascade training

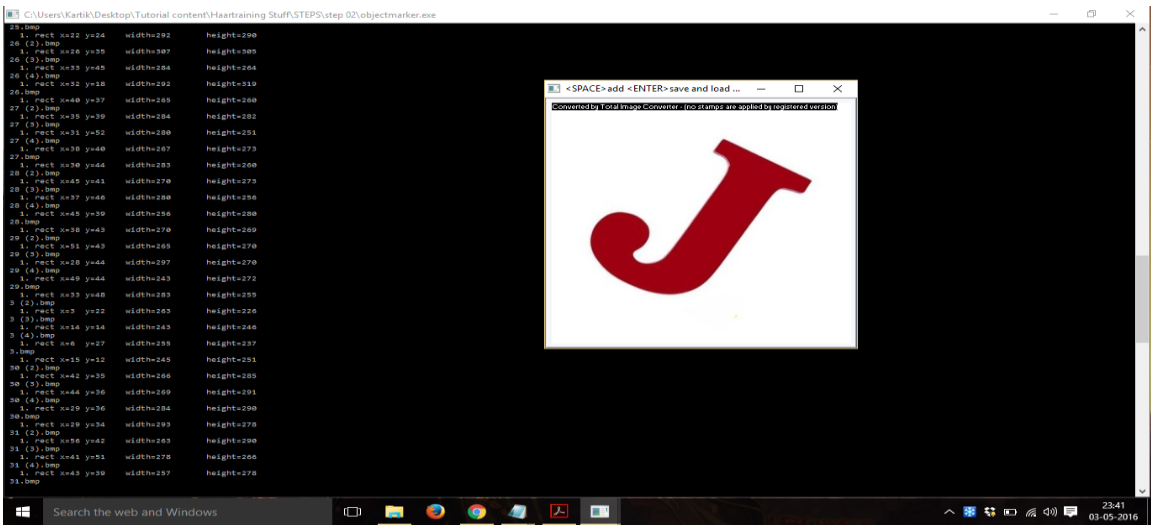


Fig 4.3:Screenshot of Cascade training

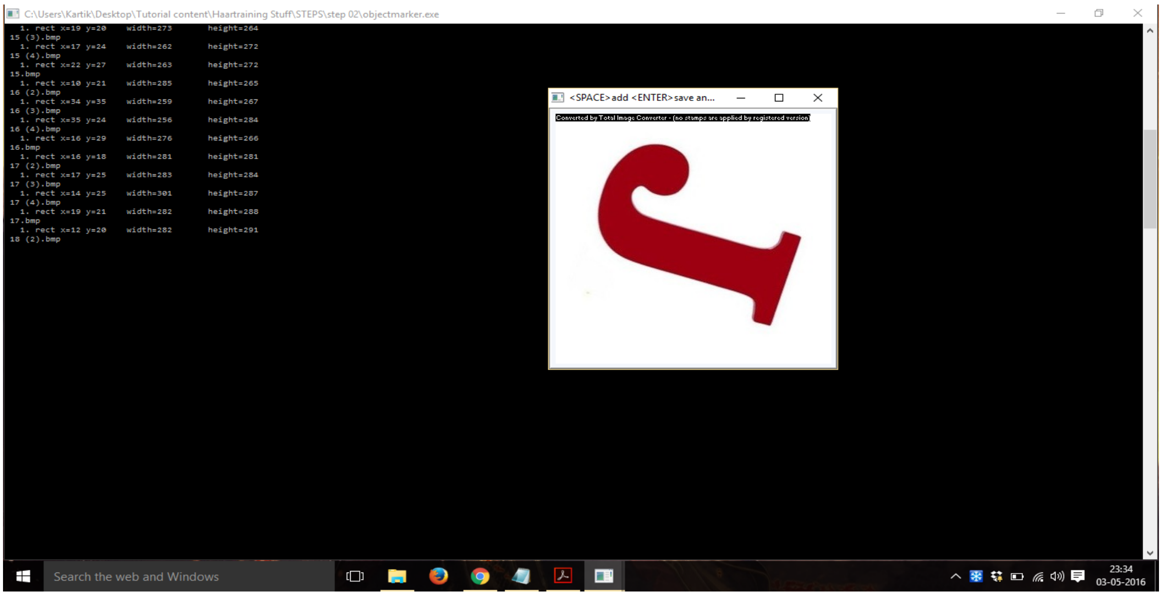


Fig 4.4:Screenshot of Cascade training

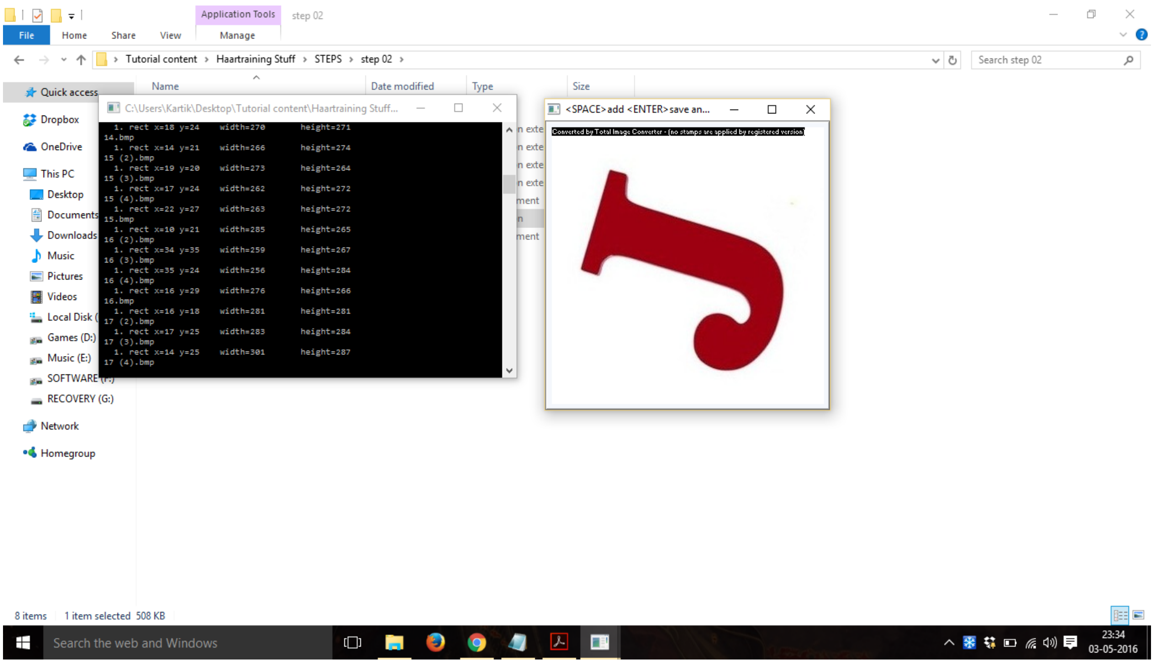


Fig 4.5:Screenshot of Cascade training

<u>Creating a vec file:</u>

opencv_createsamples –info info,lst –num 50 –w 60 –h 60 –vec positive.vec

<u>Training the classifier:</u>

opencv –traincascade –data data –vec positive.vec –bg bg.txt –numPos 50 –numNeg 25 –numStages 15 –w 50 –h 50 –featureType HAAR

4.1.2 Avoiding Object Collusion-with Ultrasonic sensor

Ultrasonic transducers are transducers that convert ultrasound waves to electrical signals or vice versa. Those that both transmit and receive may also be called ultrasound transceivers; many ultrasound sensors besides being sensors are indeed transceivers because they can both sense and transmit. These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves, respectively. Active ultrasonic sensors generate high-frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.

The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects. It essentially gives your Arduino eyes / special awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor. This simple project will use the HC-SR04 sensor with an Arduino and a Processing sketch to provide a neat little interactive display on your computer screen.

Sound consists of oscillating waves through a medium (such as air) with the pitch being determined by the closeness of those waves to each other, defined as the frequency. Only some of the sound spectrum (the range of sound wave frequencies) is audible to the human ear, defined as the “Acoustic” range. Very low frequency sound below Acoustic is defined as “Infrasound”, with high frequency sounds above, called “Ultrasound”. Ultrasonic sensors are designed to sense object proximity or range using ultrasound reflection, similar to radar, to calculate the time it takes to reflect ultrasound waves between the sensor and a solid object. Ultrasound is mainly used because it's inaudible to the human ear and is relatively accurate within short distances.

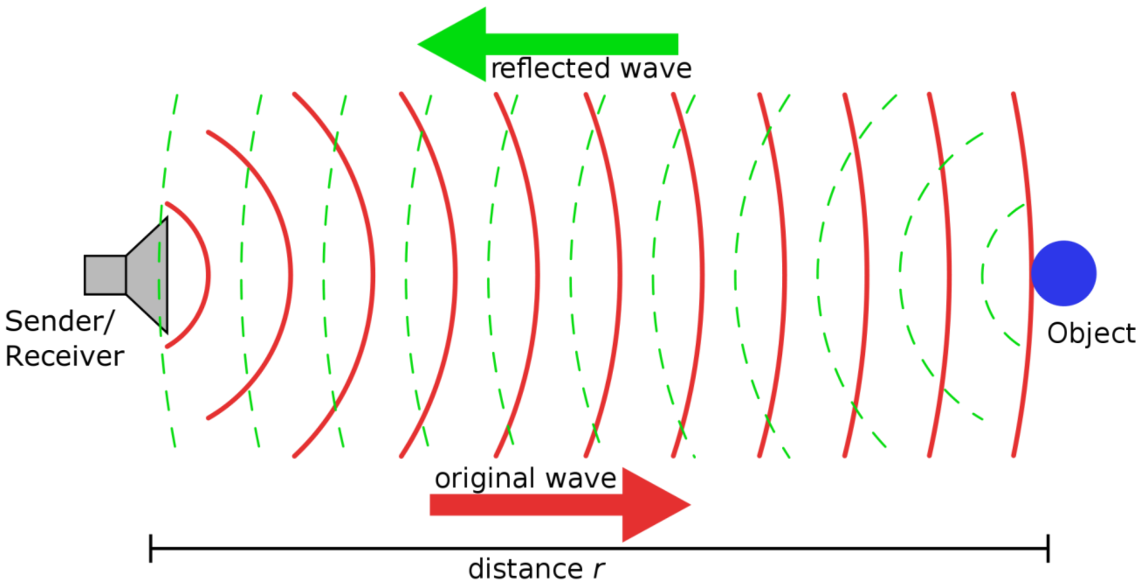


Fig 4.6:Ultrasonic Sensor Working

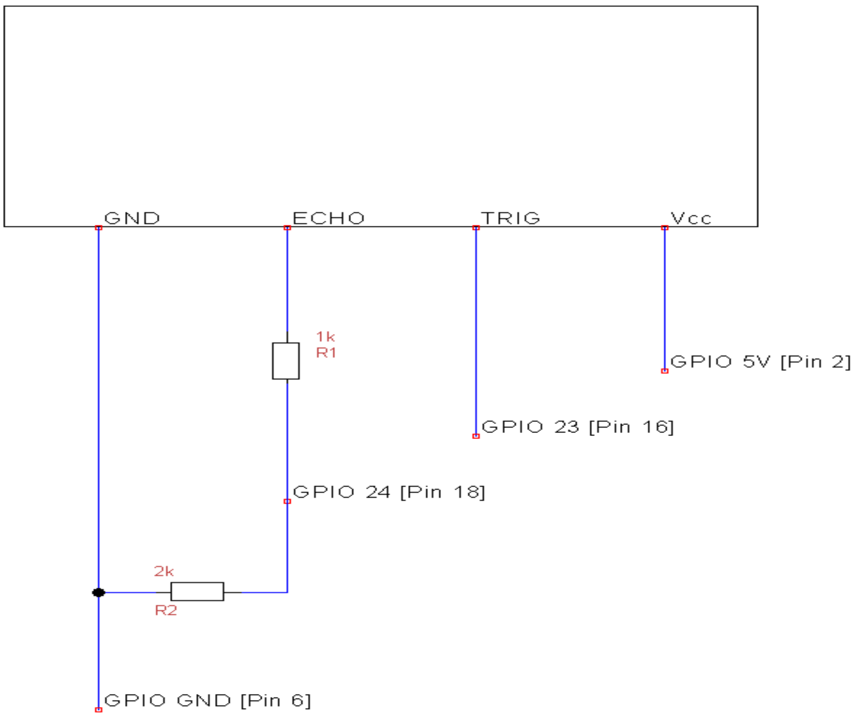


Fig 4.7:Pin-out of Ultrasonic Sensor

4.1.3 Security of the Luggage –using HCSR-501 PIR Motion Sensor

HC-SR501 is based on infrared technology, automatic control module, using Germany imported LHI778 probe design, high sensitivity, high reliability, ultra-low-voltage operating mode, widely used in various auto-sensing electrical equipment, especially for battery-powered automatic controlled products. Specification:

◦ Voltage: 5V – 20V

◦ Power Consumption: 65mA

◦ TTL output: 3.3V, 0V

◦ Lock time: 0.2 sec

◦ Trigger methods: L – disable repeat trigger, H enable repeat trigger

◦ Sensing range: less than 120 degree, within 7 meters

◦ Temperature: – 15 ~ +70

◦ Dimension: 32*24 mm, distance between screw 28mm, M2, Lens dimension in diameter: 23mm

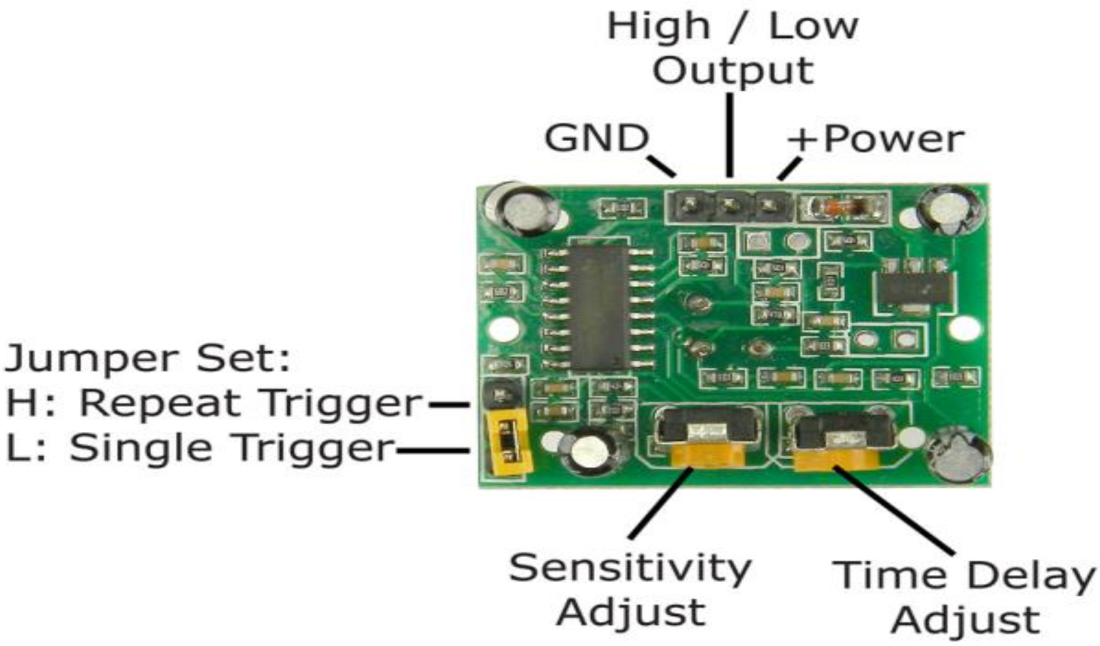


Fig 4.8:HCSR-501 PIR Motion Sensor

Features:

- Automatic induction: to enter the sensing range of the output is high, the person leaves the sensing range of the automatic delay off high, output low.
- Photosensitive control (optional, not factory-set) can be set photosensitive control, day or light intensity without induction.
- Temperature compensation (optional, factory reset): In the summer when the ambient temperature rises to 30 ° C to 32 ° C, the detection distance is slightly shorter, temperature compensation can be used for performance compensation.
- Triggered in two ways: (jumper selectable)
    - Non-repeatable trigger: the sensor output high, the delay time is over, the output is automatically changed from high level to low level;
    - Repeatable trigger: the sensor output high, the delay period, if there is human activity in its sensing range, the output will always remain high until the people left after the delay will be high level goes low (sensor module detects a time delay period will be automatically extended every human activity, and the starting point for the delay time to the last event of the time).
- With induction blocking time (the default setting: 2.5s blocked time): sensor module after each sensor output (high into low), followed by a blockade set period of time, during this time period sensor does not accept any sensor signal. This feature can be achieved sensor output time “and” blocking time “interval between the works can be applied to interval detection products; This function can inhibit a variety of interference in the process of load switching. (This time can be set at zero seconds – a few tens of seconds).
- Wide operating voltage range: default voltage DC4.5V-20V.
- Micro-power consumption: static current

4.1.4 Object Tracking and Direction –with Servo Motors

Pan[7]: Moving the camera lens to one side or another. Look to your left, then look to your right - that's panning. During a pan, the camera is aimed sideways along a straight line. Note that the camera itself is not moving. It is often fixed on tripod, with the operator turning it either left or right. Panning is commonly utilized to capture images of moving objects like cars speeding or people walking.

Direction: As we are using a 3-wheel technology, the front wheel is used for directions. The front wheel moves as the camera pans.

As the image detected the Raspberry Pi sends values to the Arduino board. Accordingly the Arduino sends values to the servo Motors and as per the values the Panning and Direction Servo motors rotate. Basically the servos are going to move the webcam and keep the subjects pattern in the center of the video feed. Pragmatically speaking, Processing takes the video input from the webcam and uses the OpenCV library to analyze the video. If a pattern is detected in the video, the OpenCV library will give the Processing sketch the coordinates of the pattern. The processing sketch will determine where the pattern is located in the frame, relative to the center of the frame, and send this data through a serial connection to an Arduino. The Arduino will use the data from the Processing sketch to move the servos connected to the Pan/Direction servos.

Basically the servos are going to move the webcam and keep the subjects pattern in the center of the video feed. Pragmatically speaking, Processing takes the video input from the webcam and uses the OpenCV library to analyze the video. If a pattern is detected in the video, the OpenCV library will give the Processing sketch the coordinates of the pattern. The processing sketch will determine where the pattern is located in the frame, relative to the center of the frame, and send this data through a serial connection to an Arduino. The Arduino will use the data from the Processing sketch to move the servos connected to the Pan/Direction bracket.

Basically this sketch will analyze a serial input for commands and set the servo positions accordingly. The command structure for the serial commands is simple. A command consists of two bytes: a servo ID and a servo position. If the Arduino receives a servo ID, then it waits for another serial byte and then assigns the received position value to the servo identified by the servo ID. The Arduino Servo library is used to easily control the pan and tilt servos. There aren't that many variables; a couple are used to keep track of the servo ID values for each servo, and then an object (or instance) is created for each servo. Finally there's a character variable that will be used to keep track of the characters that come in on the Serial port. After telling the Arduino where the servos are connected, we set the initial position of the servos to be 90 degrees; this is just so that the setup goes back to a good starting point every time the Arduino powers up. Finally, in order to use the serial port we set up the connection with the Serial.begin command

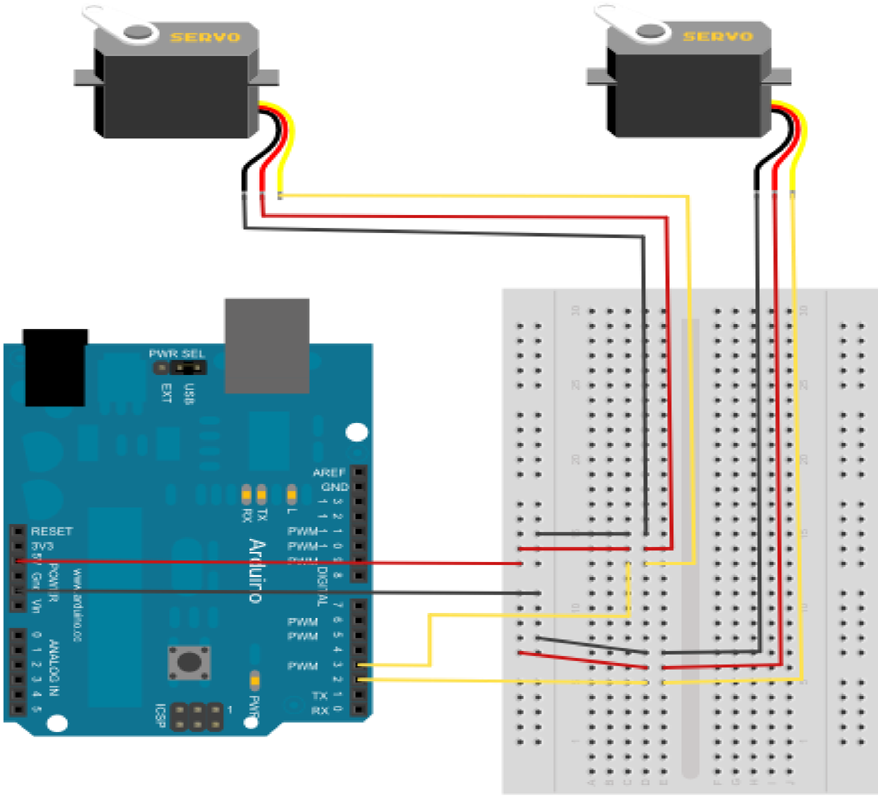


Fig 4.9:Connection of Pan/Direction Servos .

4.1.5 Movement of the Cart –using Motor shield and DC Motors

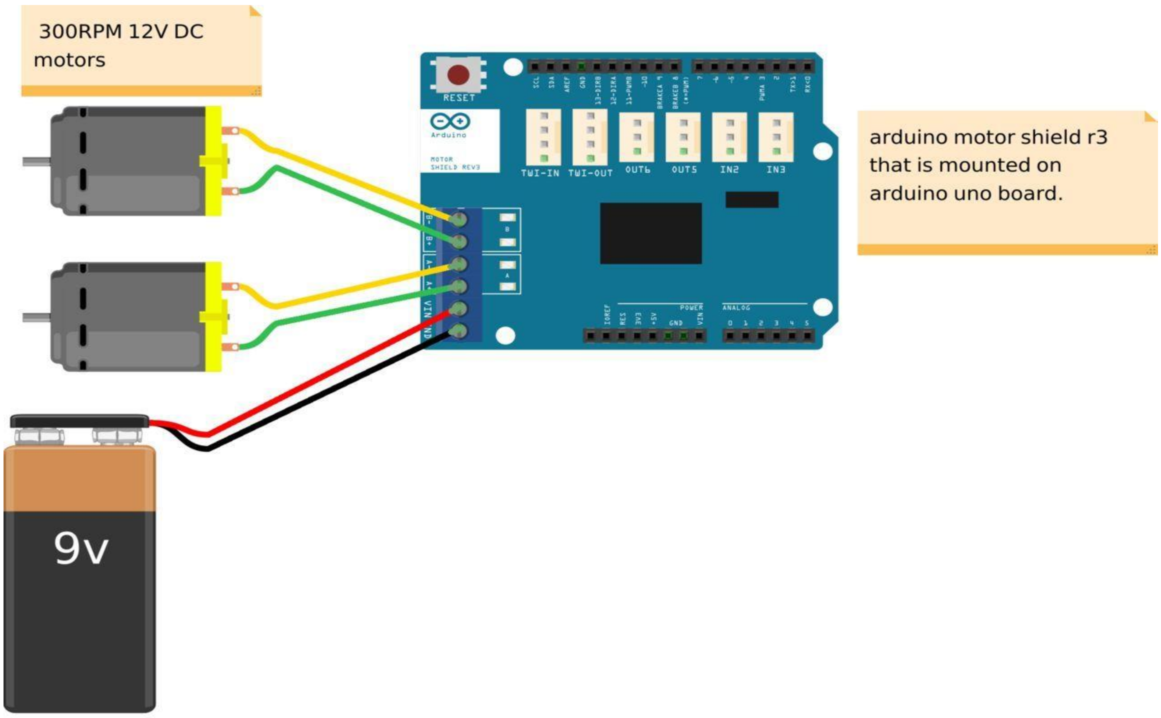


Fig 4.10:Connection of DC Motors.

The Arduino Motor Shield[8] allows you to easily control motor direction and speed using an Arduino. By allowing you to simply address Arduino pins, it makes it very simple to incorporate a motor into your project. It also allows you to be able to power a motor with a separate power supply of up to 12v. Best of all, the shield is very easy to find. The motor shield has 2 channels, which allows for the control of two DC motors, or 1 stepper motor. It also has 6 headers for the attachment of Tinker kit inputs, outputs, and communication lines. The use of these pins is somewhat limited.

With an external power supply, the motor shield can safely supply up to 12V and 2A per motor channel (or 4A to a single channel).There are pins on the Arduino that are always in use by the shield. By addressing these pins you can select a motor channel to initiate, specify the motor direction (polarity), set motor speed (PWM), stop and start the motor, and monitor the current absorption of each channel. The pin breakdown is as follows:

Function	**Channel A**	**Channel B**
Direction	Digital 12	Digital 13
Speed (PWM)	Digital 3	Digital 11
Brake	Digital 9	Digital 8
Current Sensing	Analog 0	Analog 1

Table 4.1: Pins of Motor Shield

Voltage requirements:

The first important thing to figure out what voltage the motor is going to use. If you're lucky your motor came with some sort of specifications. Some small hobby motors are only intended to run at 1.5V, but its just as common to have 6-12V motors. The motor controllers on this shield are designed to run from 5V to 12V.

Current requirements:

The second thing to figure out is how much current your motor will need. The motor driver chips that come with the kit are designed to provide up to 1.2 A per motor, with 3A peak current. Note that once you head towards 2A you'll probably want to put a heat-sink on the motor driver, otherwise you will get thermal failure, possibly burning out the chip

Setting up your shield for powering DC Motors:

The DC motors are powered off of a 'high voltage supply' and NOT the regulated 5V. Don't connect the motor power supply to the Arduino's 5V power pin. This is a very very very bad idea unless you are sure you know what you're doing! You could damage your Arduino and/or USB port!

There are two places you can get your motor 'high voltage supply' from.

1. One is the DC barrel jack on the Arduino board
2. The other is the 2-terminal block on the shield that is labeled DC Motor Power 5-12VDC.

The DC Jack on the Arduino has a protection diode so you won't be able to mess things up too bad if you plug in the wrong kind of power. The terminal block as a protection FET so you will not damage the Arduino/shield if you wire up your battery supply backwards, but it won’t work either.

Working of the DC Motors:

The DC Motor module when combined with the whole system helps in moving the cart. Once the image is detected by the camera module it sends it to the Raspberry Pi which further directs it to the Arduino which is connected serially to it.The Arduino tracks the x and y co-ordinates of the image detected ,and once this is done the Arduino sends the signals to motor shield to rotate the DC motors.

While programming the Arduino we use AFMotor.h library to send signals to the motor shield from the Arduino.

4.3 ARCHITECTURAL DESIGN

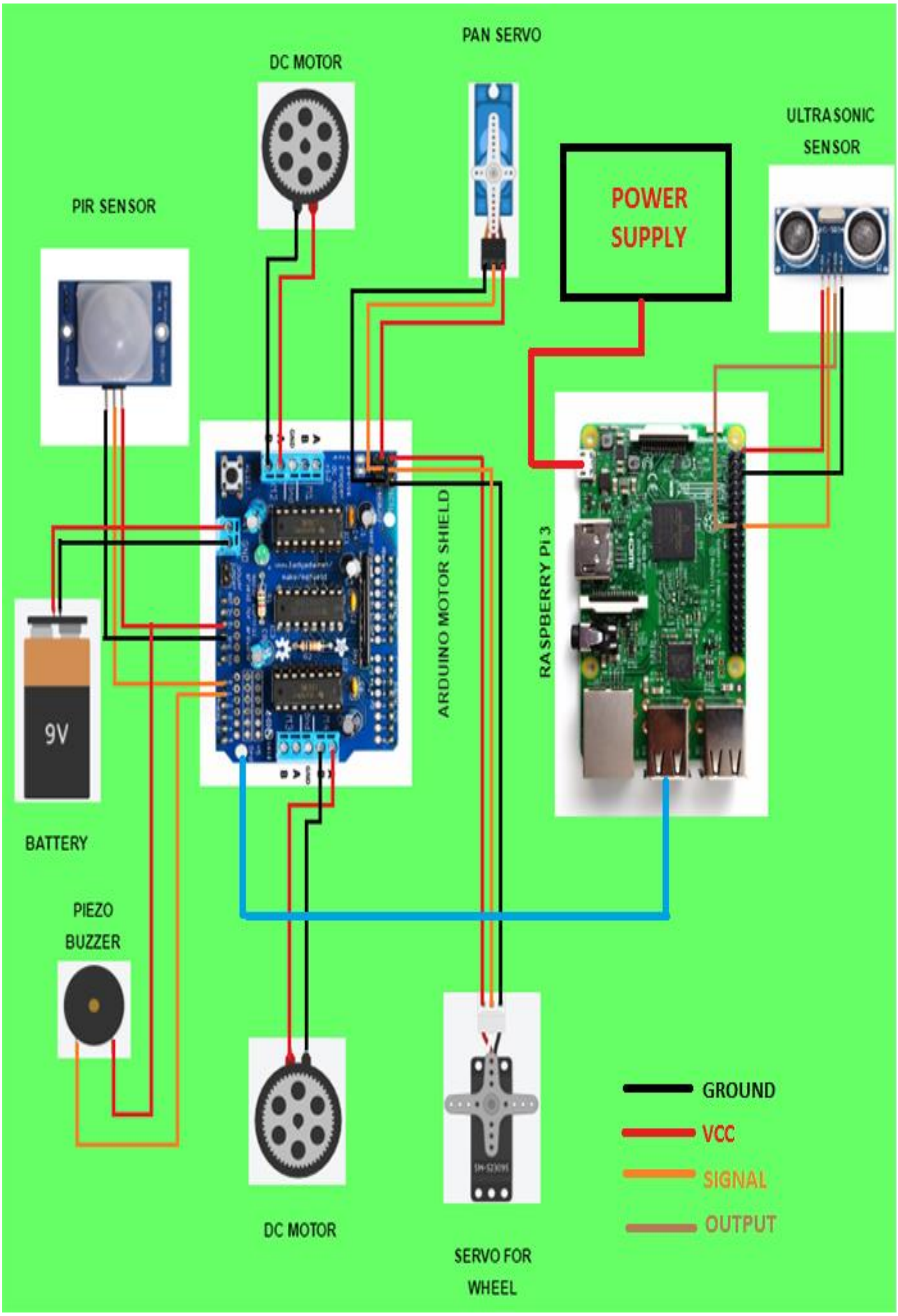


Fig 4.11 :Architectural Design

4.4 Class-Responsibility-Collaborators Diagram

Class PAN	Collaborators
Responsibility • Movement of camera • Tracking the pattern.	• Pattern • Camera

Class Camera	Collaborators
Responsibility • Detecting the pattern	• Pattern

Class PIR Sensor	Collaborators
Responsibility • Provides a mechanism which senses any motion of a warm body like human it generates a negative differential change • Send signals to Buzzer	• Buzzer

Class Buzzer	Collaborators
Responsibility • Beeps when it receives signals from PIR Sensor.	• PIR Sensor

Class Ultrasonic Sensor	Collaborators
Responsibility • Stops the Movement wheels when any object is sensed in front of it.	• DC Motors

Class Direction Wheel	Collaborators
Responsibility • Moves according to the camera	• Camera

<table>
<tr><td>Class DC Motor</td><td rowspan="2">Collaborators
Pattern
Camera
Ultrasonic sensor</td></tr>
<tr><td>Responsibility
Movement of Back wheels when the pattern is detected
Stops the Back wheels when any object is detected by Ultrasonic sensor</td></tr>
</table>

CHAPTER: 5

IMPLEMENTATION

5.1 Arduino assembly:

- Begin by placing the Arduino on a flat surface.
- Add jumper wires to all the header pins. These will be connected to the PIR Sensor and Piezo Buzzer.
- Add Motor Shield on top of the stack.
- Connect the servos to the motor shield.
- Connect the two DC motors to motor shield.
- Connect the Arduino to the computer using USB cable and upload the program from Arduino IDE.
- It is programmed such that it tracks the object by changing the servo positon which is obtained from Raspberry Pi serially.
- As, the object is detected it also drives the DC motors connected to it such that it follows the object.
- While, the object is being followed if any motion is detected it triggers the alarm for alert.

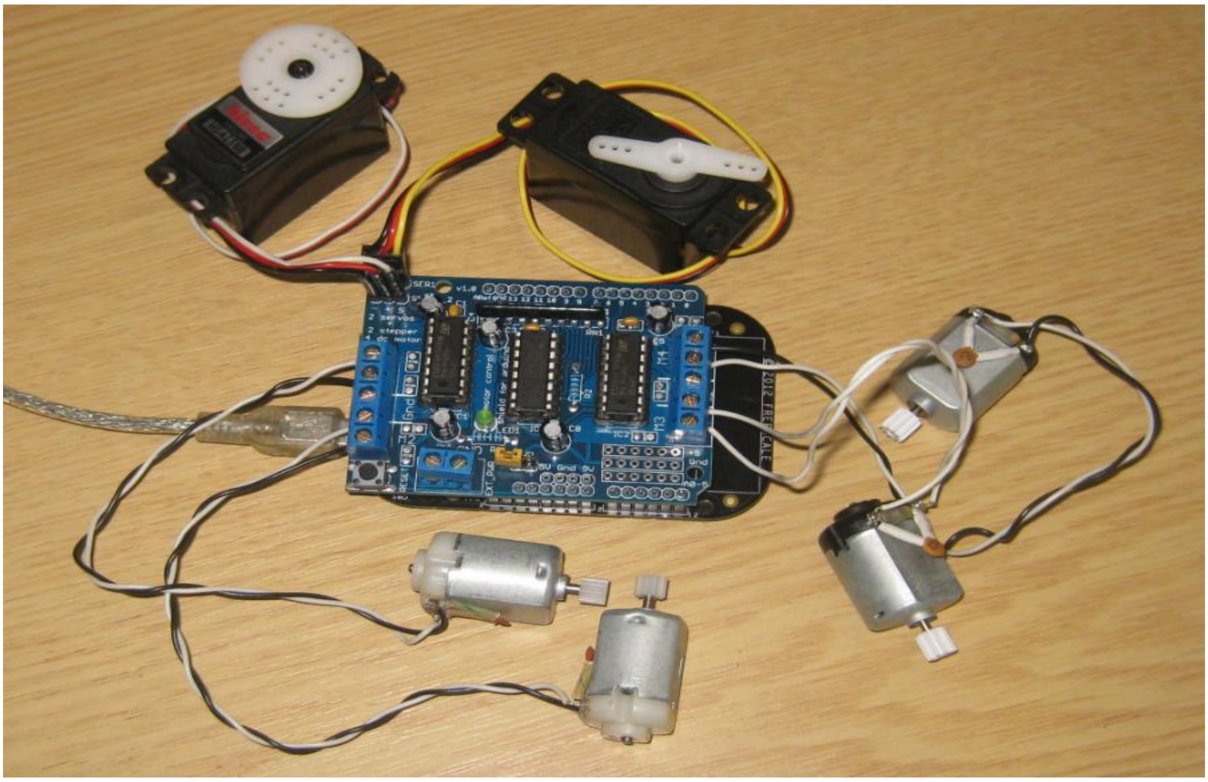


Figure 5.1:Arduino Assembly

5.2 Cascade Classifier training Opencv:

- Collect the positive images for detection. Here we are training the cascade to detect a single image bellow.

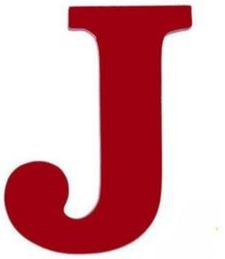


Fig 5.2:Positive image

- Collect the negative samples which should not be detected by the classifier, also called as background images. The number negative images must be greater than positive images.

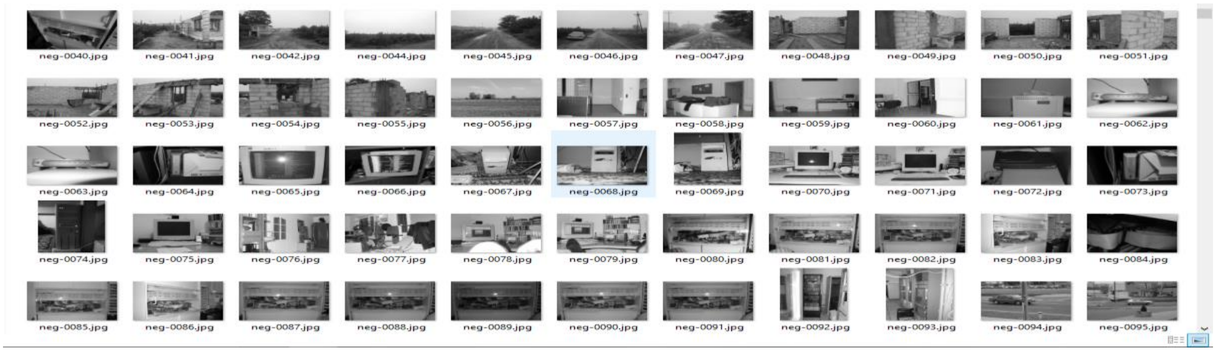


Fig 5.3: List of Negative images

- Create a text file that contains list of all negative images as shown below, by the following command:

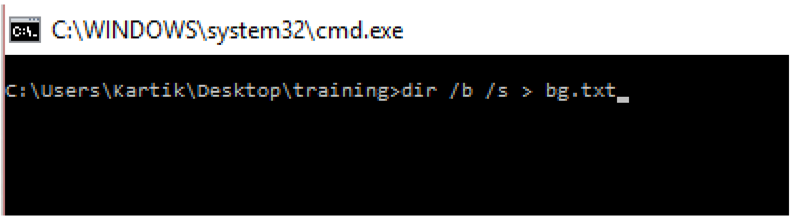
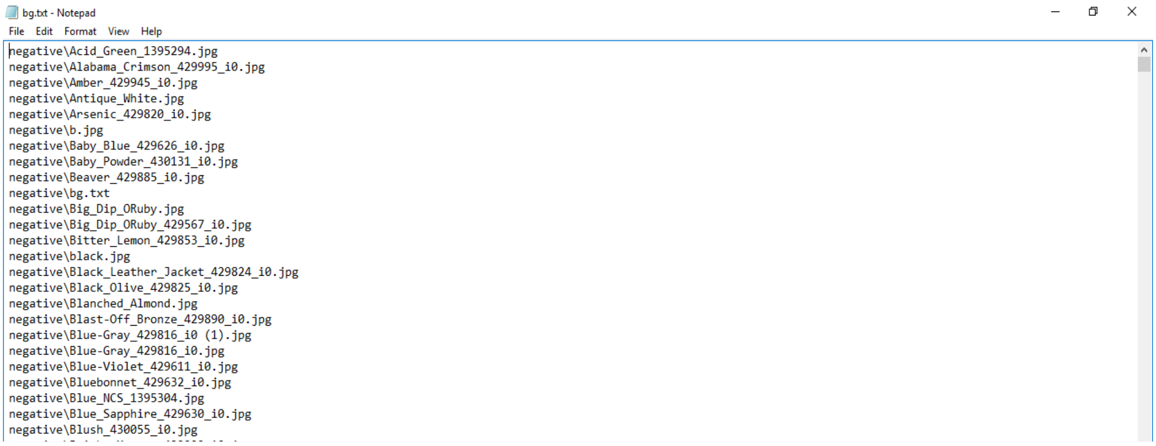


Fig 5.4: Command to create New text file



bg.txt - Notepad

```
negative\Acid_Green_1395294.jpg
negative\Alabama_Crimson_429995_i0.jpg
negative\Amber_429945_i0.jpg
negative\Antique_White.jpg
negative\Arsenic_429820_i0.jpg
negative\b.jpg
negative\Baby_Blue_429626_i0.jpg
negative\Baby_Powder_430131_i0.jpg
negative\Beaver_429885_i0.jpg
negative\bg.txt
negative\Big_Dip_ORuby.jpg
negative\Big_Dip_ORuby_429567_i0.jpg
negative\Bitter_Lemon_429853_i0.jpg
negative\black.jpg
negative\Black_Leather_Jacket_429824_i0.jpg
negative\Black_Olive_429825_i0.jpg
negative\Blanched_Almond.jpg
negative\Blast-Off_Bronze_429890_i0.jpg
negative\Blue-Gray_429816_i0 (1).jpg
negative\Blue-Gray_429816_i0.jpg
negative\Blue-Violet_429611_i0.jpg
negative\Bluebonnet_429632_i0.jpg
negative\Blue_NCS_1395304.jpg
negative\Blue_Sapphire_429630_i0.jpg
negative\Blush_430055_i0.jpg
```

Fig 5.5:List of Negative images

- Generate more number of positive samples with a single image to be detected and create a .info file b the following command:

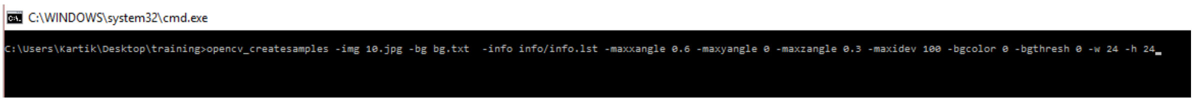
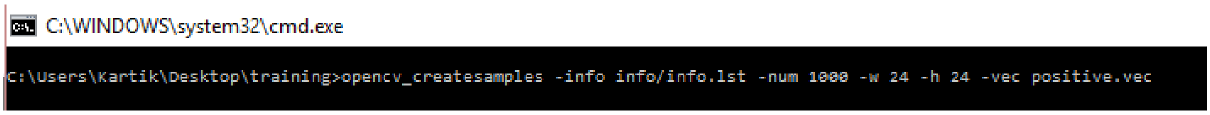
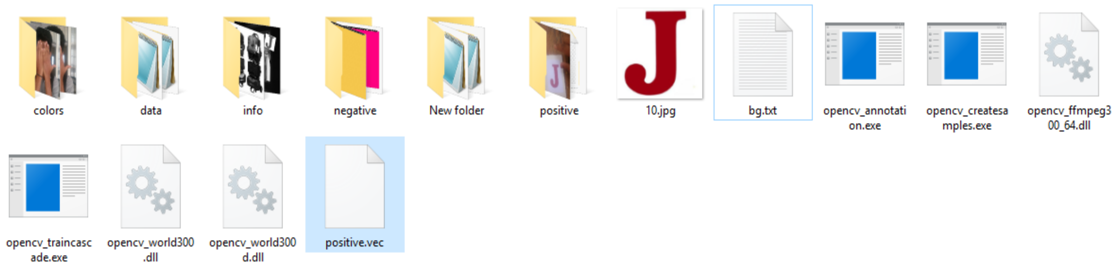


Fig 5.6:Command to create .info file

- Now generate the .vec file as shown below, where obtained image is placed onto an arbitrary background from the background description file, resized to the desired size specified by –w and –h and stored to the .vec file.





- After executing all the above commands without any errors, train the classifier as follows:

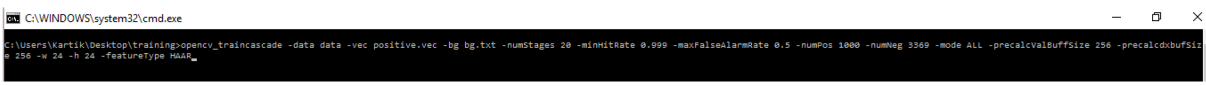


Fig 5.7: Cascade Training

- When the training is completed successfully, xml files are created which can be used in program to detect the object.

cascade.xml	29-03-2016 23:35	XML File	17 KB
params.xml	28-03-2016 22:49	XML File	1 KB
stage0.xml	28-03-2016 22:49	XML File	1 KB
stage1.xml	28-03-2016 22:51	XML File	1 KB
stage2.xml	28-03-2016 22:53	XML File	1 KB
stage3.xml	28-03-2016 22:56	XML File	1 KB
stage4.xml	28-03-2016 22:59	XML File	1 KB
stage5.xml	28-03-2016 23:02	XML File	1 KB
stage6.xml	28-03-2016 23:04	XML File	1 KB
stage7.xml	28-03-2016 23:07	XML File	1 KB
stage8.xml	28-03-2016 23:09	XML File	1 KB
stage9.xml	28-03-2016 23:11	XML File	1 KB
stage10.xml	28-03-2016 23:14	XML File	1 KB
stage11.xml	28-03-2016 23:17	XML File	1 KB
stage12.xml	28-03-2016 23:21	XML File	1 KB
stage13.xml	28-03-2016 23:27	XML File	1 KB
stage14.xml	28-03-2016 23:40	XML File	1 KB
stage15.xml	29-03-2016 00:25	XML File	1 KB
stage16.xml	29-03-2016 01:43	XML File	1 KB

Fig 5.8: Completion of Training

5.3 Raspberry Assembly:

- Place the Raspberry Pi close to Arduino.
- Connect the programmed Arduino to the Raspberry Pi using the USB cable.
- Connect the Raspberry Pi GPIO pins to pin outs of Ultrasonic sensor.
- Power on the Raspberry Pi with external battery, write the code for anti-object collision that gets the values from the ultrasonic sensor, such that whenever if any object is too close to the cart it must stop.
- Program the raspberry pi that detects the object.
- When the object is detected it sends the X and Y co-ordinates to the Arduino which tracks the object by moving the servo position as the object is moved.
- Similarly, whenever the object is detected the DC motors starts and follows the object.

CHAPTER: 6

TESTING

6.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole.

6.2 Functional Testing

Functional testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional testing usually describes what the system does.Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.

Functional testing basically the testing of the functions of component or system is done. It refers to activities that verify a specific action or function of the code. Functional test tends to answer the questions like “can the user do this” or “does this particular feature work”.

6.3 Performance Testing

Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device. This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions. Qualitative attributes such as reliability, scalability and interoperability may also be evaluated. Performance testing is often done in conjunction with stress testing.

Performance testing can verify that a system meets the specifications claimed by its manufacturer or vendor. The process can compare two or more devices or programs in terms of parameters such as speed, data transfer rate, bandwidth, throughput, efficiency or reliability.

6.4 Integration testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing.

6.5 Acceptance testing

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

There are various forms of acceptance testing:

- User acceptance Testing
- Business acceptance Testing
- Alpha Testing

Chapter 7:

RESULT ANALYSIS

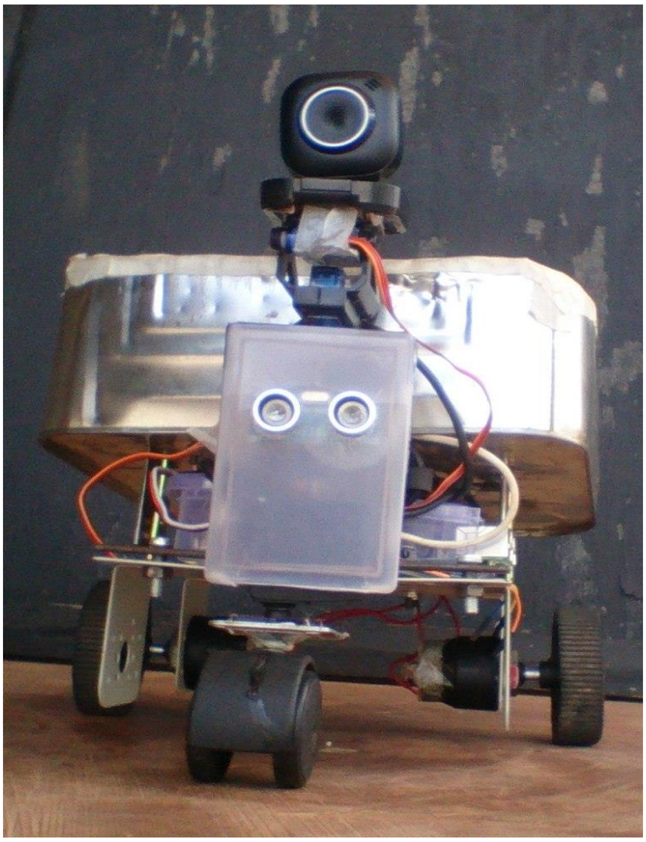


Fig 7.1 Front-View of Cart

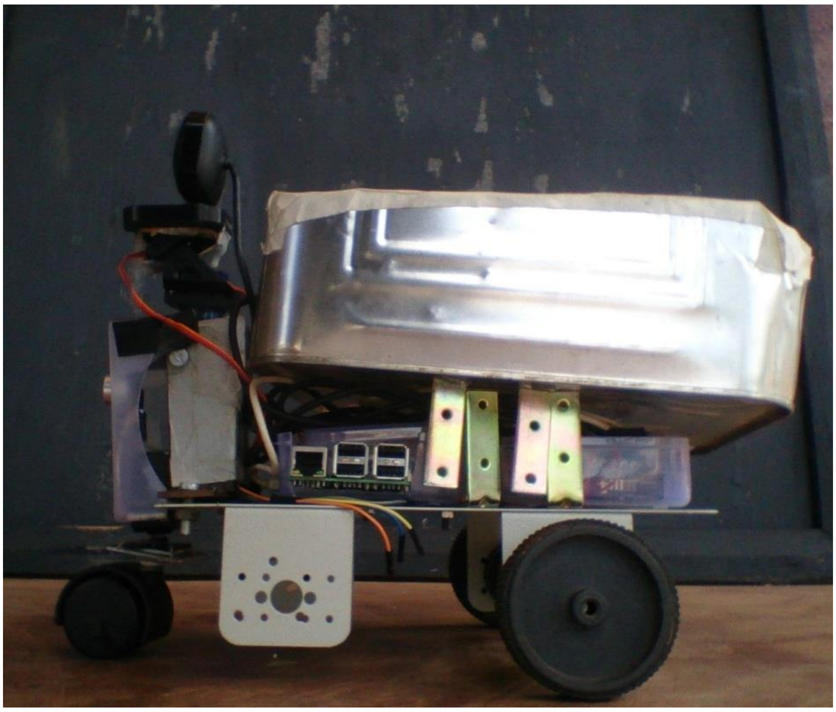


Fig 7.2 Side-View of Cart

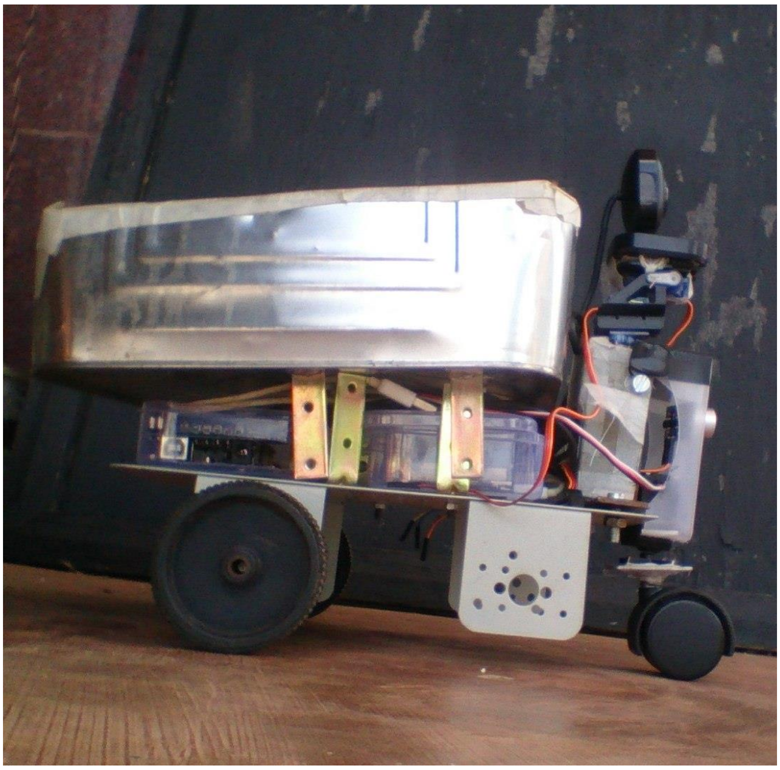


Fig 7.3 Side-View of Cart

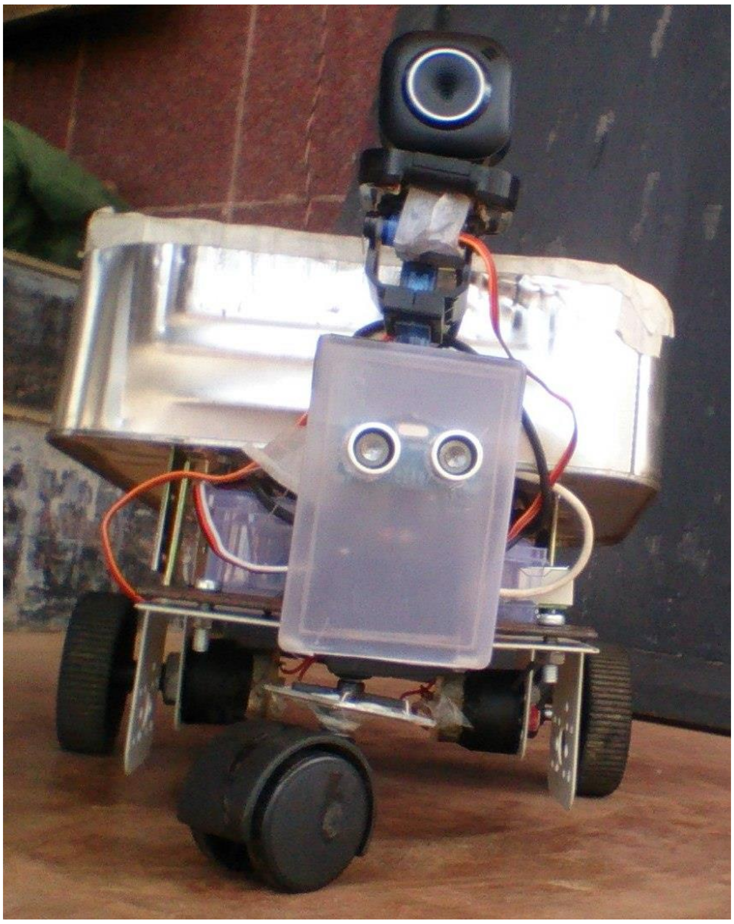


Fig 7.4 Front-View with change in direction wheel

CONCLUSION AND FUTURE WORK

Smart luggage carrier is a very unique ,innovative and creative approach to carry luggage’s without any man power with the help of beautiful technology .smart luggage carrier not only helps the user to carry heavy luggage’s but also provides security to the luggage’s. Smart luggage carrier is easy to use and user friendly. The user can trust smart luggage carrier based on its performance, reliability and the capabilities it provides the user of the cart. image processing with Opencv and using of sensors to provide convince to the user to make it a superior one.

It can used in airport, malls etc. It can be used in smart city plan. It can be used to scan the luggages and minimize the terrorist activities. In future the cart can be connected to the cell phone so that the user can track his luggage and make it more secure.

.

REFERENCES

[1] Md. Imran Khan, Saad Bin Siddique, Nazmul Hassan, Md.Towhid Chowdhury Automated Luggage Carrying System 2013 from : URL: http://www.ajer.org/papers/v2(11)/G02116170.pdf

[2]Open-CV Python Tutorials

URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

[3] Raspberry Pi details

URL : https://www.arduino.cc/en/main/arduinoBoardUno

[4]Arduino Uno details

URL: https://www.raspberrypi.org/

[5] Cascade classifier training

URL: http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html#training-data-preparation

[6] Open-CV Image Detection

URL http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html

[7] Pan Toturials

URL: https://www.sparkfun.com/tutorials/304

[8] Motor Shield for Arduino

URL: https://learn.adafruit.com/adafruit-motor-shield/faq