1.1 Introduction

Robotics has become a very common application in most of the developed countries. High performance, high accuracy, lower labor cost and the ability to work in hazardous places have put robotics in an advantageous position over many other such technologies.

Robotics can be defined as the science or study of the technology primarily associated with the design, fabrication, theory, and application of robots, [6]. While other fields contribute the mathematics, the techniques, and the components, robotics creates the magical end product. The practical applications of robots drive development of robotics and drive advancements in other sciences in turn. Crafters and researchers in robotics study more than just robotics.

Robotic vision continues to be treated including different methods for processing, analyzing, and understanding. All these methods produce information that is translated in to decisions for robots. From start to capture images and to the final decision of the robot, a wide range of technologies and algorithms are used like a committee of filtering and decisions.

A robotic vision system has to make the distinction between objects and in almost all cases has to tracking these objects. Applied in the real world for robotic applications, these machine vision systems are designed to duplicate the abilities of the human vision system using programming code and electronic parts. As human eyes can detect and track many objects in the same time, robotic vision systems seem to pass the difficulty in detecting and tracking many objects at the same time.

A robotic system finds its place in many fields from industry and robotic services. Robots understand the environment condition and it can perform more complex tasks better. Major advances of object recognition stand to revolutionize Artificial Intelligence (AI) and robotics

Gathered visual data from cloud robotics can allow multiple robots to learn tasks associated with object recognition faster. Robots can also reference massive databases of known objects and that knowledge can be shared among all connected robots. Scientists at

Brigham Young University have developed an object recognition algorithm that can learn to identify objects on its own. The Evolution-Constructed Features algorithm, as

it's called, can make decisions about what characteristics of an object are relevant to its identification.

Concerns about the potential of object recognition include fears that advertisers and other interested entities will use the technology to mine the increasing number of images posted online and gather from them the personal information of individuals.

Robot Object Recognition: The human visual system is extremely powerful. It is equipped with a high selectivity that allows us to distinguish among even very similar objects, like the faces of identical twins. Some studies believe that the human visual system can discriminate among at least tens of thousands of different object categories.

An Object Identifier is a name used to identify an object. This object can be a country or an individual disk drive. The most common one, in the context of the IEEE-RAC (Registration Authority Committee), is the OUI (Organizationally Unique Identifier), and the organizationally derived, and assigned, assignments beyond the OUI. Most common subsequent identifiers include Ethernet address identifier, the Extended Unique Identifiers (EUI) or the World Wide Name (WWN) identifiers.

1.2 Objective of the Project

Today, robotics is one of the best technologies which deal with design, working and applications of Robots, computer systems for their control and information processing. This technology also deals with automated machines and is very useful in manufacturing process.

The main aim of this project is to design an autonomous harvesting robot which can detect and pluck the fruits or vegetables from the tree. In this project, a robot is designed in such a way that the robot moves automatically towards the tree. Whenever it detects the one of the color from RGB then the robot plucks the fruit or vegetable.

1.3 Literature Survey

.Third world countries are still not very familiar with the use of robots. There are two reasons behind this. Firstly, the high initial costs in importing robots and associated software and secondly, conceptions of the capabilities of robots as day to day essential applications.

Fire Bird V will help you get acquainted with the world of robotics and embedded systems. Thanks to its innovative architecture and adoption of the 'Open Source Philosophy' in its software and hardware design, you will be able to create and contribute too, complex

applications that run on this platform, helping you acquire expert is easy out to spend more time with them. Fire Bird V is designed by NEX Robotics and Embedded Real-Time Systems lab, CSE IIT Bombay.

As a Universal Robotic Research Platform, Fire Bird V provides an excellent environment for experimentation, algorithm development and testing. Fire Bird V is evolved from Fire Bird V and Fire Bird II which are being used in IIT Bombay to teach embedded systems and robotics.

Its modular architecture allows you to control it using multiple processors such as 8051, AVR, PIC and ARM7 etc. Modular sensor pods can be mounted on the platform as dictated by intended applications.

1.4 Organization of the Project report

There are five chapters in this report. Introduction about project is presented in chapter 1. Chapter 2 describes existing methods. Chapter 3 describes about the Arduino UNO, installation of Arduino software, Arduino software usage and interfacing modules of harvesting robot. Chapter 4 describes results of the project and Chapter 5 describes conclusion and future scope.

2.1 Introduction

For manual method, fruits are harvested by cutting them off with pruning shears, clippers or by pulling the fruit stalk from the tree.

The manual method includes hand picking, (where harvesting of citrus fruits is done by pulling or clipping from the stem). The general rule is **"**twist, jerk, and pull**"**. This procedure is highly recommended for tight-skinned citrus like the Late Valencia, [7]. This method is used when the fruit is within a reachable height.

Fig. 2.1: Twist, jerks, and pulls method

However, the method is not recommended for harvesting tangerines as the skin around the (stem end) will pull away from the fruit, [8]. The fruit will then be open for infection and will deteriorate rapidly. To prevent damage, clip the fruit off by cutting the stem just above the button. Lemons to be stored should also be clipped off in the same manner. Also the use of simple tools such as the clippers and ladder can be classified under the manual method of harvesting citrus fruits.

Fig. 2.2: Harvesting fruit with ladder and bag

Also the use of simple tools such as the clippers and ladder can be classified under the manual method of harvesting citrus fruits.

The ladder is used to scaffold the one doing the cutting to a height where he or she cactus or detach the fruit from the tree. In place of the ladder other machines (power ladders) can be used to achieve same results. Power ladders are used to harvest fruit that cannot be reached from ground level. The clipper's blades are designed and shaped in such a way that they will fit into the calyx of the fruit, cutting the stem of as close to the calyx as possible.

2.2 Manual methods

2.2.1 Shake the limb and the fruit falls off

This first technique is the quickest and easiest. All you need are a few willing bodies and a large sheet under the fruit tree. Pick a fruit-filled branch and give it a good shake. This method is useful for tall trees and works best for those whose fruit can fall off easily, such as apples. Most important, the mass quantity of fruit is collected gently and safely. Of course, when shaking the tree, everything comes down, including the leaves and so forth, so you'll have to sort out the good fruit from the bad fruit, leaves and other debris.

Fig. 2.3: Shake the limb and the fruit falls off

2.2.2 Use a fruit-picker basket

For hard-to-reach fruits that can be pulled off easily, use a fruit-picker basket that has an extension handle. You can wrap the hooks around the fruit's stem and give it a tug. The fruit falls right into the basket as shown in Fig. 2.4.

2.2.3 Prune off fruit

Use handheld pruners to remove fruits that are hard to remove by pulling. For those hard-to-reach tender fruits, employ telescoping pole pruners. As an added bonus, pole pruners hold as they cut, so this is perfect for soft and fragile fruit like ripe peaches.

Fig. 2.4: Harvesting fruits by fruit-picker basket

2.2.4 Twist and pull

Some fruits like pomegranates don't come off the tree as easily as apples. If you don't have pruners handy, grab each individual fruit and twist it until it's free of the tree. When harvesting is complete, inspect the fruit, picking out the bad ones to be composted, [8]. If you have extra fruit and want to donate them, call your local food bank or do some research online for drop-off locations and times for any local harvest services.

Fig. 2.5: Twist and pull method of harvesting fruits

2.3 Advantages of Manual Harvesting

- Accurate maturity and grade selection.
- Easy to increase harvest rate-just hire more pickers.
- Visual image processing ability which enables workers rapidly to detect fruit suitable for harvest and direct their hand to the fruit selected for detachment.
- Less damage to product.
- Allows multiple harvests.
- They can chose to pick only the ripen fruits, or rather, they can avoid picking rotten or bad fruits.

2.4 Disadvantages of Manual Harvesting

- Manual harvest is also a slow and painstaking process. Unless you have a lot of money, so that you can afford to hire a really big army of pickers.
- It is having long process of manually plucking of fruits
- Low capacity, it is expected that much of the world's fruit will continue to be harvested by hand for the foreseeable future.
- It contains the shortage of skilled labor.
- It is danger for the labor personnel from thorns& reptiles in the gardens etc. are the major hurdle for the fruit growers, [9].

2.5 Difference between Manual harvesting and Robotic harvesting

- Speed up the process of harvesting in case of robot compared to manual.
- The robot able to work effectively in any climate condition (high temperature, fog etc.) rather than humans.
- Fruits are getting damaged by manual plucking and this is not occurred in robot plucking.

.

3.1 Introduction

The robotics designers offer to the farmers the opportunity to significantly reduce the costs of manual labor for harvesting. This robot can replace the seasonal or even permanent employees on farm.

The harvesting robot is designed to detect, recognize and determine the ripe fruit to be picked. In addition, the robot able to harvest the fruits without damaging them.

We have designed an automated harvesting robot that will move throughout the fruit garden and reach all areas of the tree to pluck the fruits or vegetables. A color detection sensor is used to detect the primary color fruits. Whenever the robot senses the colors (RGB) then robotic arm activate to pluck the fruit from the tree.

3.2 Block Diagram

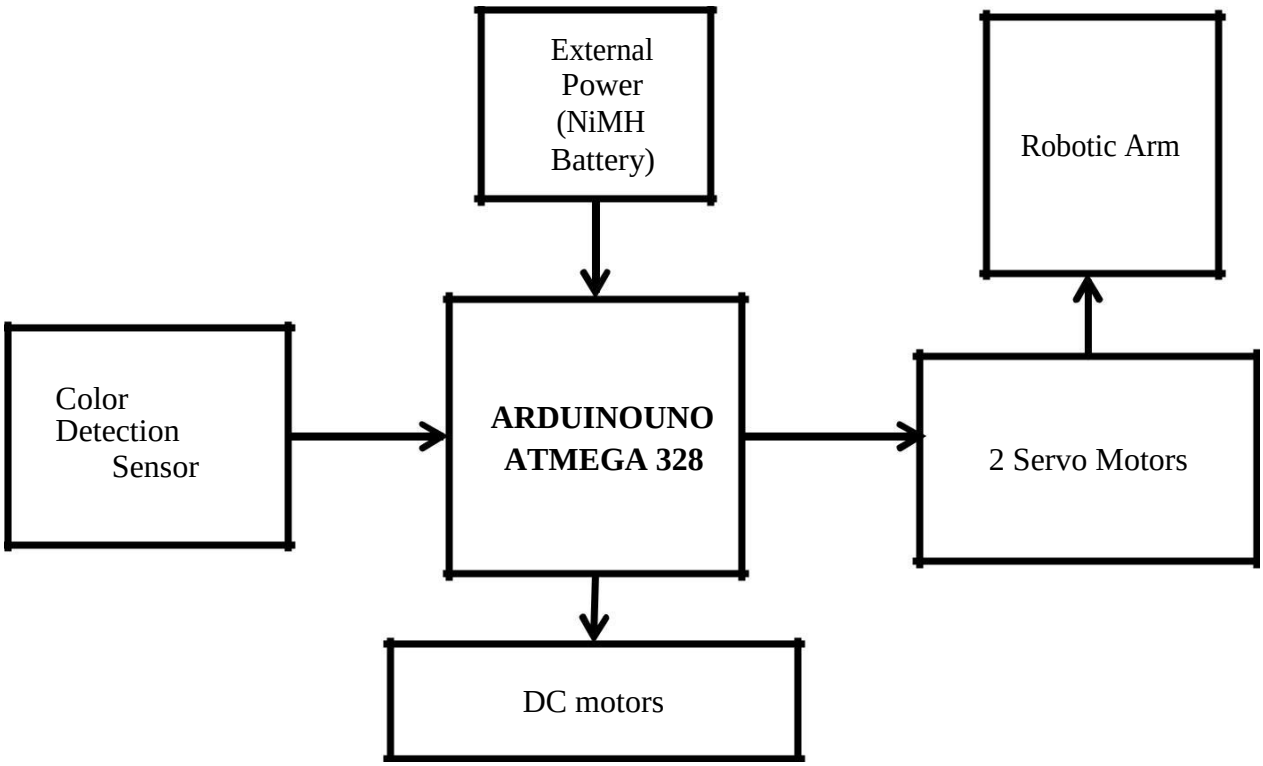

Fig. 3.1: Block Diagram of Harvesting Robot

The major parts of the harvesting robot include Arduino UNO (ATMEGA 328 Microcontroller), Robotic arm, Servo motors, Color detection sensor and Gear DC motors. Now we are briefly describing about each module and it's interfacing with Arduino.

3.3 ARDUINO UNO

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, 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, [10].

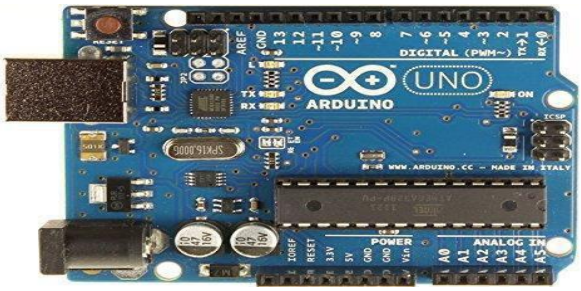

Fig. 3.2: Arduino UNO

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward.

3.3.1 History of Arduino UNO

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003 Hernando Barragán created the development platform wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non- engineers.

The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller. In 2003, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino.

The initial Arduino core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis, but Barragán was not invited to participate.

Following the completion of the Wiring platform, lighter and less expensive versions were distributed in the open-source community.

Adafruit Industries, a New York City supplier of Arduino boards, parts, and assemblies, estimated in mid-2011 that over 300,000 official Arduino’s had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

In October 2016, Federico Musto, Arduino's former CEO, secured a 50% ownership of the company. In April 2017, Wired reported that Musto had "fabricated his academic record.... On his company's website, personal LinkedIn accounts, and even on Italian business documents, Musto was until recently listed as holding a PhD from the Massachusetts Institute of Technology. In some cases, his biography also claimed an MBA from New York University." Wired reported that neither University had any record of Musto's attendance, and Musto later admitted in an interview with Wired that he had never earned those degrees.

Around that same time, Massimo Banzi announced that the Arduino Foundation would be "a new beginning for Arduino. “But a year later, the Foundation still hasn't been established, and the state of the project remains unclear.

The controversy surrounding Musto continued when, in July 2017, he reportedly pulled many Open source licenses, schematics, and code from the Arduino website, prompting scrutiny and outcry.

In October 2017, Arduino announced its partnership with ARM Holdings (ARM). The announcement said, in part, "ARM recognized independence as a core value of Arduino... without any lock-in with the ARM architecture.” Arduino intends to continue to work with all technology vendors and architecture.

3.3.2 Arduino Features

Some of the key features of Arduino include

i An open source design. The advantage of it being open source is that it has a large community of people using and troubleshooting it. This makes it easy to

find someone to help you debug your projects.

ii An easy USB interface. The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.

iii Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.

iv A 16 MHz clock. This makes it not the speediest microcontroller around, but fast enough for most applications.

v 32 KB of flash memory for storing your code.

vi 13 digital pins and 6 analog pins. These pins allow you to connect external hardware to your Arduino. These pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these pins and you are good to go.

vii An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to re-boot load your chip if it corrupts and can no longer talk to your computer.

viii An on-board LED attached to digital pin 13 for fast an easy debugging of code

ix And last, but not least, a button to reset the program on the chip.

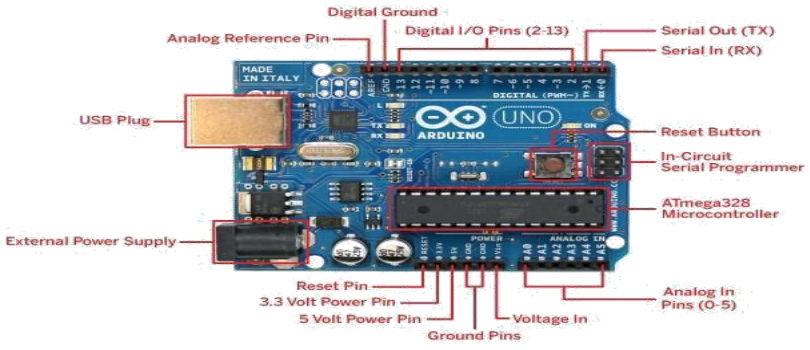

Fig. 3.3: Arduino pin description

The heart of Arduino UNO board is ATMEGA 328 Microcontroller. AMEGA 328 Microcontroller is used to program and function the Arduino board for a specific application.

3.3.3 ATMEGA 328 Microcontroller

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega328/P provides the following [11] features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 1 serial programmable USARTs , 1 byte-oriented 2-wire Serial Interface (I2C), a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages) , a programmable Watchdog Timer with internal Oscillator, an SPI serial port, and six software selectable power saving modes.

Fig. 3.4: ATMEGA328 Microcontroller

The Idle mode stops the CPU while allowing the SRAM; Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset.

In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This

allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In- System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core.

The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega328/P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega328/P is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, and Program Debugger/Simulators, In- Circuit Emulators, and Evaluation kits. The technical specifications of ATMEGA 328 Microcontroller are given in below table.

Table. 3.1: Technical Specifications of ATMEGA 328

Features	**ATMEGA 328**
Pin Count	28/32
Flash (Bytes)	32K
SRAM(Bytes)	2K
EEPROM(Bytes)	1K
Interrupt Vector Size	1/12
General Purpose I/O Lines	23
SPI	2
TWI(I2C)	1
USART	1
ADC	10-bit 15kSPS
ADC Channels	8
8-bit Timer/Counters	2
16-bit Timer/Counters	1

There are 28 pins in ATMEGA 328 Microcontroller. Description of pins is given in detailed manner.

i **VCC:** Digital supply voltage.

ii **GND:** Ground.

iii **Port B (PB [7:0]) XTAL1/XTAL2/TOSC1/TOSC2:** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if g the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

iv Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

v Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

vi If the Internal Calibrated RC Oscillator is used as chip clock source, PB [7:6] is used as TOSC [2:1] input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

vii **Port C (PC [5:0]):** Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC [5:0] output buffers have symmetrical drive characteristics with both high sink and source capability, [12]. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

viii **PC6/RESET:** If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is un programmed, PC6 is used as a Reset input. A low level

on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a Reset.

ix The various special features of Port C are elaborated in the Alternate Functions of Port C section.

x **Port D (PD [7:0]):** Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

xi **AVCC:** AVCC is the supply voltage pin for the A/D Converter, PC [3:0], and PE [3:2]. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC [6:4] use digital supply voltage, VCC.

xii **AREF:** AREF is the analog reference pin for the A/D Converter.

xiii **ADC [7:6] (TQFP and VFQFN Package Only):** In the TQFP and VFQFN package, ADC [7:6] serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

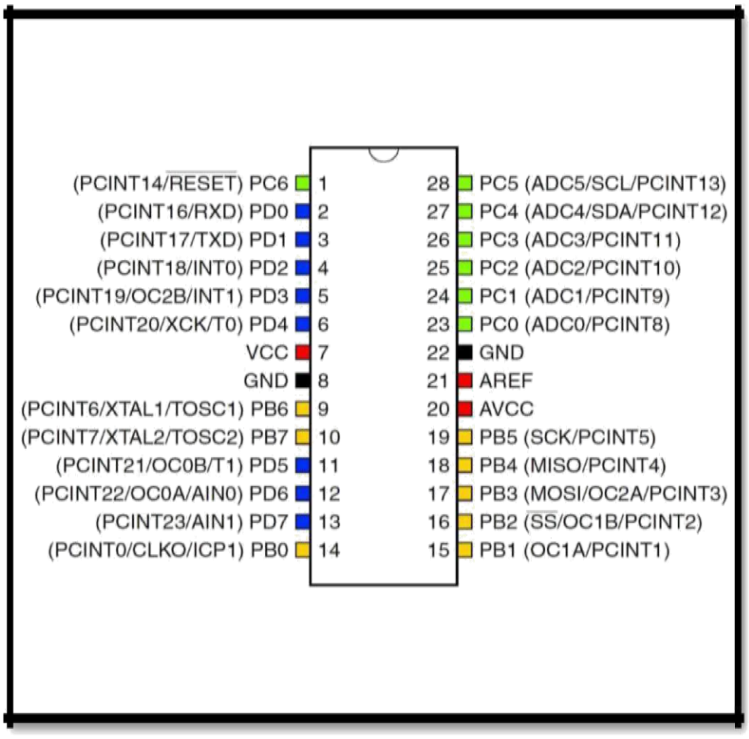

Fig. 3.5: pin-out of ATMEGA 328

3.3.4 Technical Specifications

Table. 3.2: Technical Specifications of Arduino UNO

Operating Voltage	5V
Input Voltage(recommended)	7-12V
Input voltage(limits)	6-20V
Digital I/O Pins	14
Analog I/O Pins	6
DC Current per I/O Pins	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	32 KB
SRAM	4 KB
EEPROM	1 KB
Clock Speed	16 MHz

3.3.5 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:

i **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.

ii **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.

iii **3.3V:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50mA.

iv **GND:** Ground pins.

3.3.6 Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the boot loader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

3.3.7 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

i **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of theATmega8U2

ii **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. See the attachInterrupt () function for details.

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

iv **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.

v **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 provides 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 analog Reference () 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:

i **AREF:** Reference voltage for the analog inputs. Used with analog Reference ().

ii **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.

3.3.8 Communication

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)

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-to serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1)

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

3.3.9 Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes pre burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

3.4 Installation steps of Arduino Software and Drivers

After downloading, locate the downloaded file on the computer and extract the

folder from the downloaded zipped file. Copy the folder to a suitable place such as your desktop, [13].

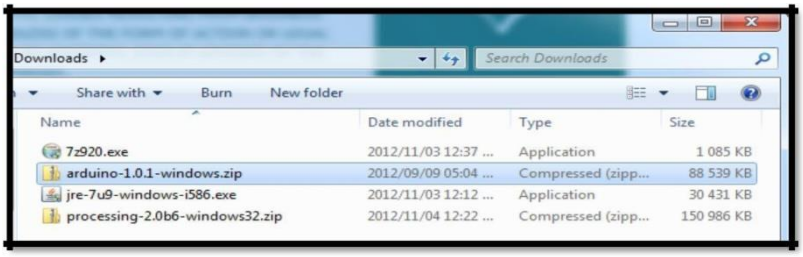

Fig. 3.6: Location of a zipped file

1. **Install the Arduino Window Drivers**

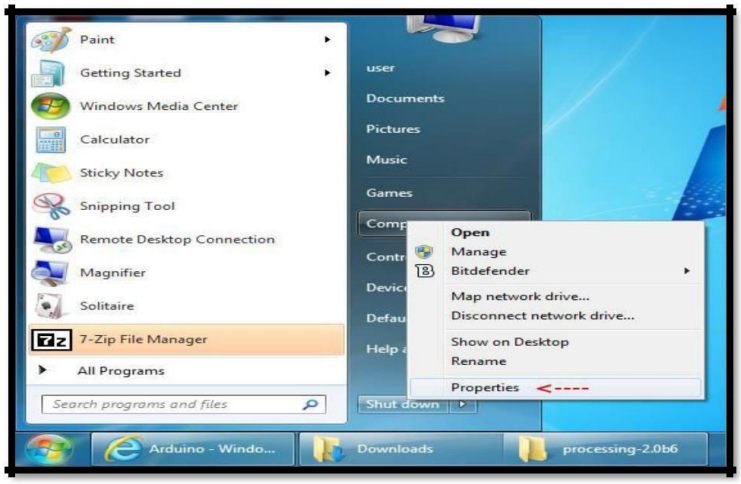

Fig. 3.7: Opening properties to Install Drivers

2. **The Device Manager will open and display the Arduino Uno**

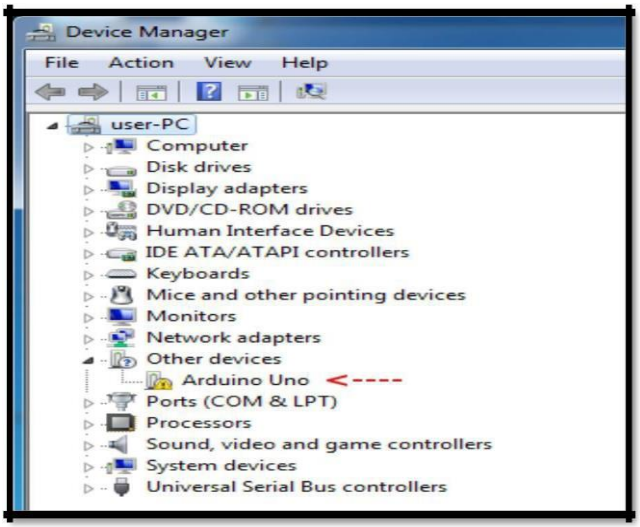

Fig. 3.8: Arduino Uno location in Device Manager

3. **Install the device driver**

a. In the Device Manager Window, right-click the Arduino board and then click **Update Driver Software...** on the pop-up menu:

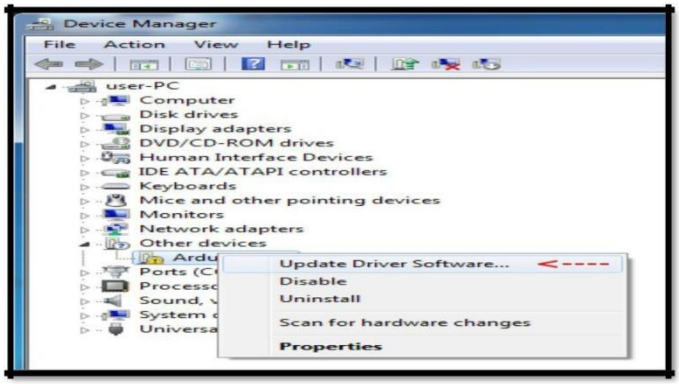

Fig. 3.9: Update Driver Software window

b. After selecting the driver folder, click the **Next** button

Fig. 3.10: Browsing for Driver Software location

4. **Setting up the Arduino Software**

a. The setup will only need to be done once, unless you change the board type or port that the Arduino is connected to.

b. Navigate to the folder that you downloaded and start the Arduino software IDE by double- clicking the Arduino application:

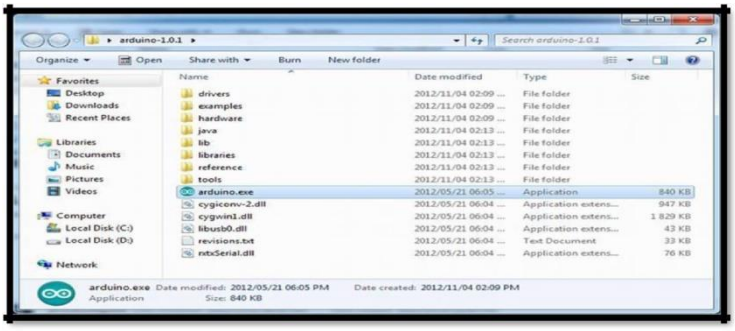

Fig. 3.11: Starting of Arduino IDE

5. **Installing of Arduino**

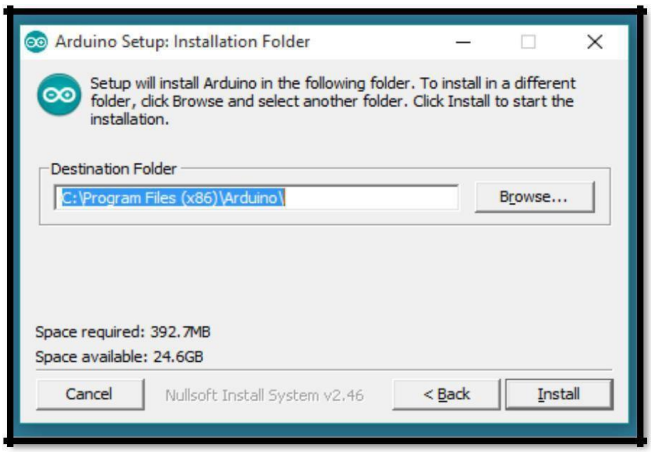

Fig. 3.12: Installing of Arduino Software

6. **Opening of Arduino software**

Fig. 3.13: Opening view of Arduino

7. **Arduino software Window**

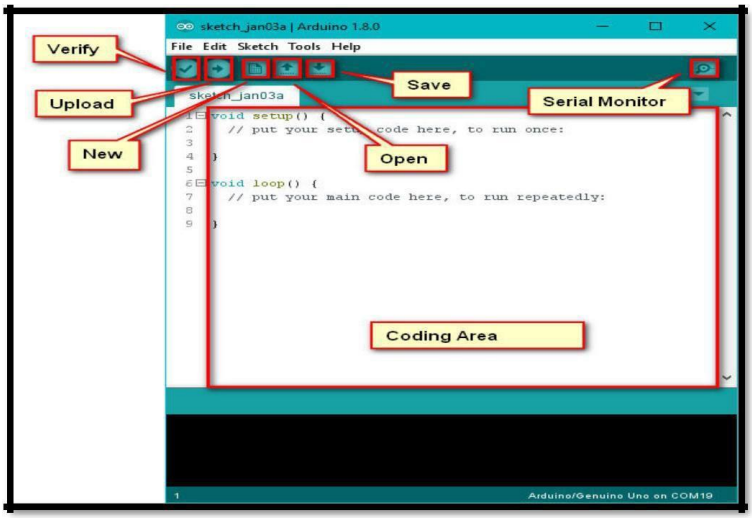

Fig. 3.14: Toolbar view of Arduino

3.5 Arduino software (IDE) usage

Step 1: Open the Arduino software

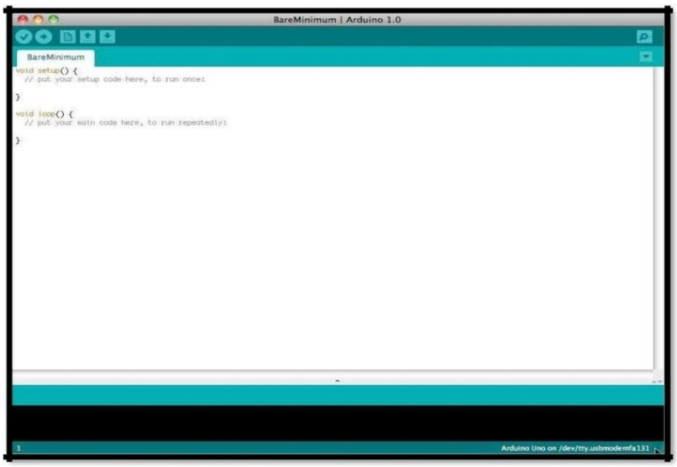

Fig. 3.15: Arduino Software Overview

Step 2: Connect the Arduino to your computer's USB port

Fig. 3.16: Connecting Arduino to PC

Please note that although the Arduino plugs into your computer, it is not a true USB device. The board has a special chip that allows it to show up on your computer as a virtual serial port when it is plugged into a USB port. This is why it is important to plug the board in. When the board is not plugged in, the virtual serial port that the Arduino operates upon will not be present (since all of the information about it lives on the Arduino board), [14-16].

It is also good to know that every single Arduino has a unique virtual serial port address. This means that every time you plug in a different Arduino board into your computer, you will need to reconfigure the serial port that is in use.

Step 3: Setting the board type and serial port

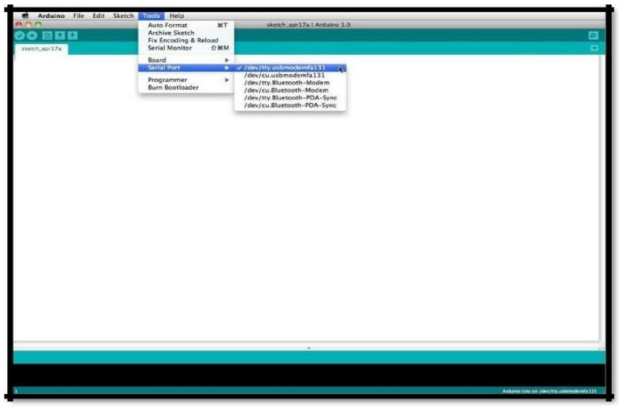

Fig. 3.17: Selecting Board type and Serial port

Step 4: selecting the programme

The blink example can be found here:

Files --> Examples --> Basics --> Blink

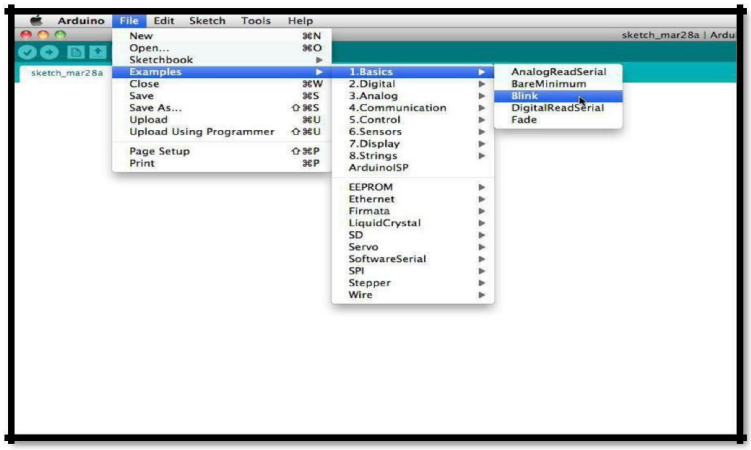

Fig. 3.18: Selecting an example

Step 5: Testing the program

Let us consider a simple example as LED blinking. The blink example basically sets pin D13 as an output and then blinks the test LED on the Arduino board on and off every second.

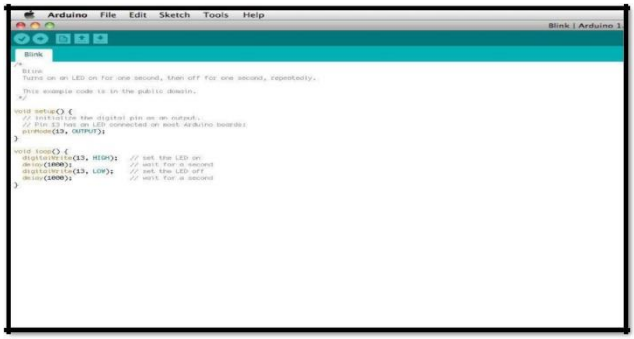

Fig. 3.19: Testing the program

Step 6: Uploading the programme

Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing the upload button, which looks like an arrow pointing to the right.

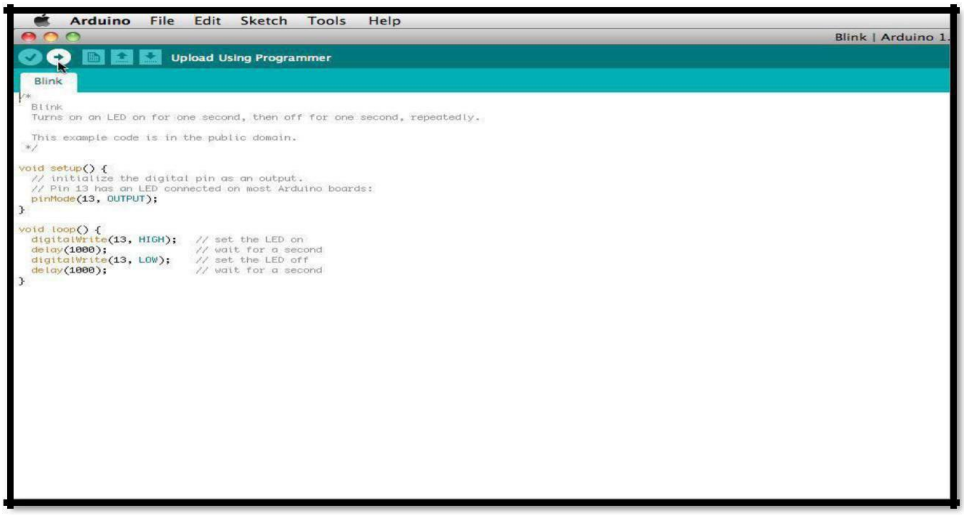

Fig. 3.20: Uploading the Program

Step 7: Checking the output in serial monitor

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displays it in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.

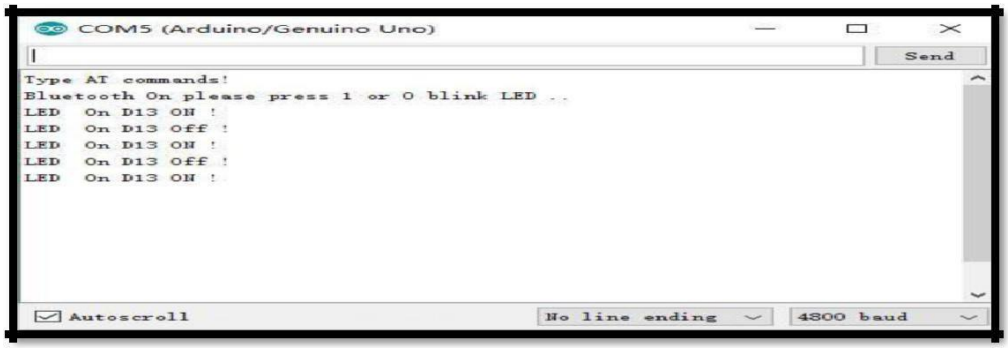

Fig. 3.21: Checking the output

3.6 Color Detection Sensor

This is a simple color sensor using Arduino Uno R3 and TCS3200 color sensor module. It can be useful for color identification and detection for food-processing units,

color printer applications, paint-mixing applications and other industrial applications including robotics.

This project is used for detecting primary colors (red, green and blue, or RGB)—colors that are physically available in LEDs in one package; for example, common cathode or common-cathode RGB LED.

We can display primary colors and also generate specific colors by modifying the Arduino code. The project demonstrates the basic interfacing of TCS3200 sensor, Arduino Uno and common-cathode RGB LED.

Fig. 3.22: TCS3200 color sensor module

TCS3200 color sensor module (SEN0101) is shown in Fig. 3.22 and microscopic view of the RGB arrays.. On the microscopic level, you can see the square boxes inside the eye on the sensor. These square boxes are arrays of the RGB matrix. Each of these boxes contains three sensors: one each for sensing red light, green light and blue light intensity. It is better than TCS230 color sensor module. This sensor can be used to identify any number of colors with an accurate programming code.

TCS3200 module has eight [18] pins. This module consists of programmable color light-to-frequency converters that combine configurable silicon photodiodes and current-to- frequency converter on a single monolithic CMOS integrated circuit. Output is square-wave (50 per cent duty cycle) with frequency directly proportional to light intensity (irradiance).

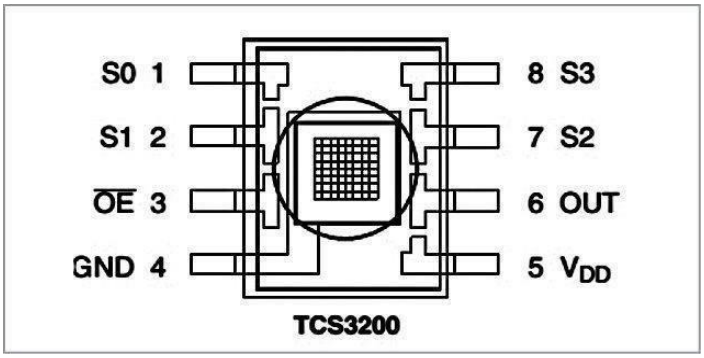

Fig. 3.23: Pin diagram of the TCS3200 color sensor module

Digital inputs and outputs allow direct interface to the MCU or other logic circuitry. Output enable (OE) places the output in high-impedance state for multiple units sharing an MCU input line. In TCS3200, the light-to-frequency converter reads an 8×8 array of photodiodes. Sixteen photodiodes have blue filters, another sixteen have green, yet another sixteen have red and remaining sixteen are clear with no filters, [17].

All photodiodes of the same color are connected in parallel. Pins S2 and S3 of TCS3200 are used to select the group of photodiodes (red, green, blue and clear) that are active.

Table. 3.3: S2 and S3 Functions

S2	S3	Photodiode Type
L	L	Red
L	H	Blue
H	L	Clear(no filter)
H	H	Green

The color which needs to be sensed by the color sensor is selected by two pins S2 and S3. With these two pins logic control we can tell sensor which color light intensity is to be measured. Say we need to sense the RED color intensity we need to set both pins to LOW.

Once that is done the sensor detects the intensity and sends the value to the control system inside the module. The light intensity measured by array is sent to current to frequency converter. What it does is, it puts out a square wave whose frequency is in relation to current sent by ARRAY.

So we have a system which sends out a square wave whose frequency depends on light intensity of color which is selected by S2 and S3. The signal frequency sent by module can be modulated depending on use. We can change the output signal frequency bandwidth.

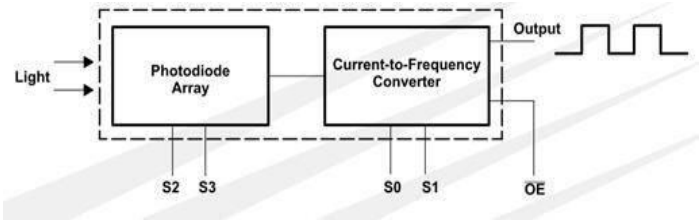

Fig. 3.24: Control System Inside the module

Table. 3.4: S0 and S1 Functions

S0	S1	Output Frequency Scaling(f_0)
L	L	Power Down
L	H	2%
H	L	20%
H	H	100%

The frequency scaling is done by two bits S0 and S1. For convenience we are going to limit the frequency scaling to 20%. This is done by setting S0 to high and S1 to LOW. This feature comes in handy when we are using the module on system with low clock.

3.6.1 Interfacing Arduino UNO with TCS3200Sensor

A color detection sensor is used to detect the primary color fruits. The below schematic shows the interfacing connections.

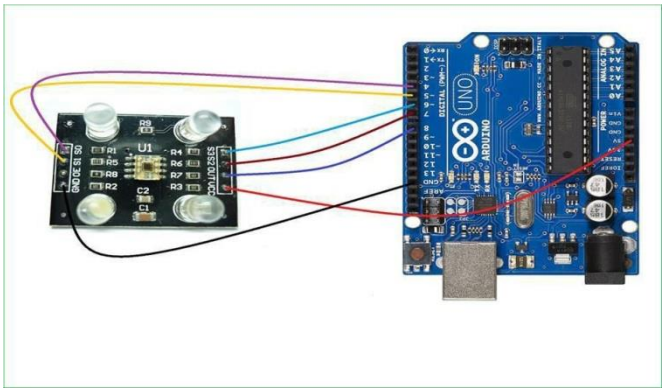

Fig. 3.25: Interfacing Schematic diagram of TCS3200 and Arduino UNO

This TCS3200 sensor module can be easily interfaced with Arduino development board, Connect the OUT pin of sensor to Arduino digital pin D8 and connect S2, S3 to Pin D7,D6 by the way connect S1,S0 to Pin D5,D4 finally connect bias to the sensor Vcc to 5V

And Gnd to Gnd pins, thats all after that upload the following Arduino code for color sensor TCS3200.

The TCS230 senses color light with the help of an 8 x 8 array of photodiodes. Then using a Current-to-Frequency Converter the readings from the photodiodes are converted into a square wave with a frequency directly proportional to the light

intensity. Finally, using the Arduino Board we can read the square wave output and get the results for the color.

If we take a closer look at the sensor we can see how it detects various colors. The photodiodes have three different color filters. Sixteen of them have red filters, another 16 have green filters, another 16 have blue filters and the other 16 photodiodes are clear with no filters.

Each 16 photodiodes are connected in parallel, so using the two control pins S2 and S3 we can select which of them will be read. So for example, if we want to detect red color, we can just use the 16 red filtered photodiodes by setting the two pins to low logic level according to the table.

The sensor has two more control pins, S0 and S1 which are used for scaling the output frequency. The frequency can be scaled to three different present values of 100 %, 20 % or 2%. This frequency-scaling function allows the output of the sensor to be optimized for various frequency counters or microcontrollers.

First we need to define the pins to which the sensor is connected and define a variable for reading the frequency. In the setup section we need to define the four control pins as outputs and the sensor output as an Arduino input.

This program uses the functions pinmode(), digitalwrite(), and delay(), which are provided by the internal libraries included in the IDE environment.

The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

The program is usually loaded in the Arduino by the manufacturer. Here we also need to set the frequency- scaling, for this example I will set it to 20%, and start the serial communication for displaying the results in the Serial Monitor, [19, 20].

In the loop section, we will start with reading the red filtered photodiodes. For that purpose we will set the two control pins S2 and S3 to low logic level. Then using the **“**pulseIn()**”** function we will read the output frequency and put it into the variable **“**frequency**”**. Using the Serial.print() function we will print the result on the serial monitor. The same procedure goes for the two other colors, we just need to adjust the control pins for the appropriate color.

Table. 3.5: Interfacing Connections of Arduino UNO and TCS3200 Sensor

Color detection sensor pins	Arduino Uno pins
S0 pin	pin D4
S1 pin	pin D5
S2 pin	pin D7
S3 pin	pin D6
Vcc	5V
GND	GND
Out pin	pin D6
OE pin	GND

3.7 Servo Motors

Servo Motors are used where there is a need for accurate shaft movement or position. These are not proposed for high speed applications. These are proposed for low speed, medium torque and accurate position application. These motors are used in robotic arm machines, flight controls and control systems, [21].

Servo motors are available at different shapes and sizes. A servo motor will have mainly there wires, one is for positive voltage another is for ground and last one is for position setting. The RED wire is connected to power, Black wire is connected to ground and YELLOW wire is connected to signal.

Fig. 3.26: Servo Motor

A servo motor is a combination of DC motor, position control system, gears. The position of the shaft of the DC motor is adjusted by the control electronics in the servo, based on the duty ratio of the PWM signal the SIGNAL pin.

3.7.1 Interfacing Arduino UNO with servo motor

A servo motor will have mainly there wires, one is for positive voltage another is for ground and last one is for position setting. The RED wire is connected to power, Black wire is connected to ground and YELLOW wire is connected to signal.

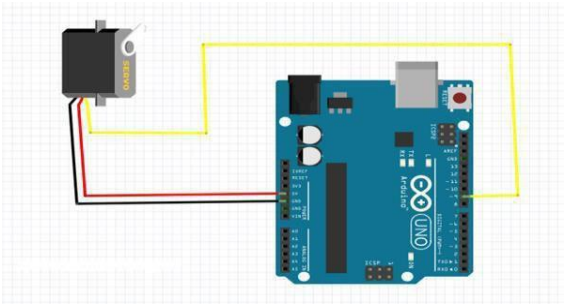

Fig. 3.27: Schematic Diagram of Arduino UNO with Servo motor

Table. 3.6: Interfacing Connections of Arduino UNO and Servo Motor

Servo Motor	**Arduino UNO**
Red Wire	5V
Yellow Wire	D10
Black Wire	GND

Simply speaking the control electronics adjust shaft position by controlling DC motor. This data regarding position of shaft is sent through the SIGNAL pin. The position data to the control should be sent in the form of PWM signal through the Signal pin of servo motor.

The frequency of PWM (Pulse Width Modulated) signal can vary based on type of servo motor. The important thing here is the DUTY RATIO of the PWM signal. Based on this DUTY RATION the control electronics adjust the shaft. For the shaft to be moved to 9o clock the TURN ON RATION must be 1/18.ie. 1ms of ON time and 17ms of OFF time in a 18ms signal.

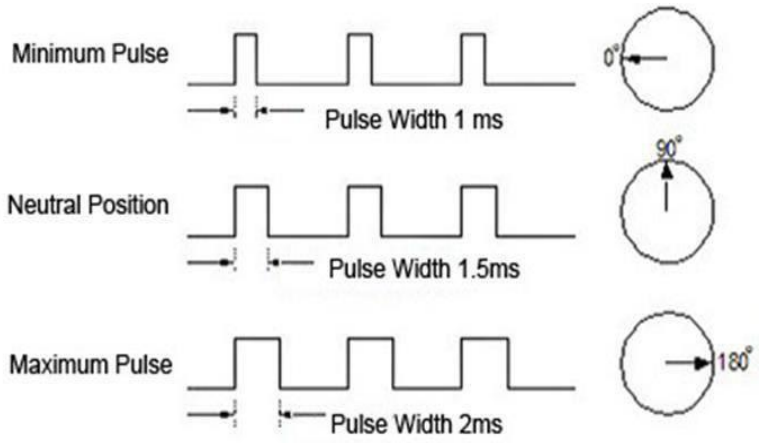

Fig. 3.28: PWM Pulses for Servo Motor

For the shaft to be moved to 12o clock the ON time of signal must be 1.5ms and OFF time should be 16.5ms. This ratio is decoded by control system in servo and it adjusts the position based on it. This PWM in here is generated by using ARDUINO UNO, [22-24].

3.8 Robotic Arm

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. The links of such a manipulator are connected by joints allowing either rotational motion (such as in an articulated robot) or translational (linear) displacement. The links of the manipulator can be considered to form a kinematic chain.

The terminus of the kinematic chain of the manipulator is called the end effectors and it is analogous to the human hand. It can perform a wide variety of functions and perform and perform the tasks where human can't do. In space as well as in vacuum metallic parts especially those which are subjected to friction specially gears and Barings wear-out very fast, [25-27].

The robotic arm can handle the particles in the chemical laboratories and used it for security purpose. It can also helpful in the medical applications. Through programming arm can handled the whole automatic industrial process. Due to the lightweight of the structure and designed technique the robotic arm can use for underwater applications in the underwater submarine. The four most advantage of this design could be very high speed manipulation, snippy manipulation due to extremely

high weight design and very low inertia.

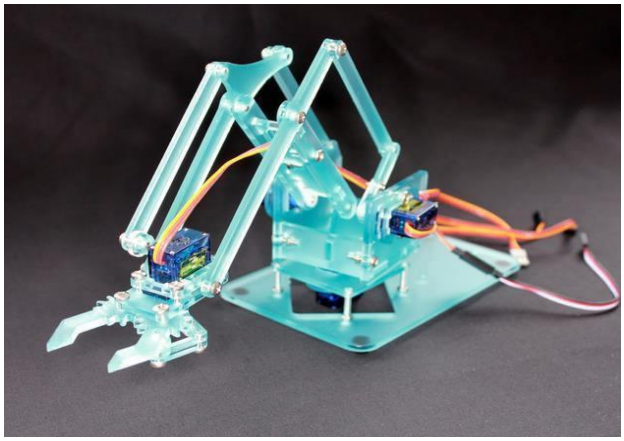

Fig. 3.29: Robotic Arm

3.8.1 Background

The word robot was derived from Czech word Robot which means **"**a forced labor**"**. Thus the robot technology is advancing rapidly. Now a day**'**s the most commonly used robots in industry is robotic hand or a robotic manipulator. Robotic hand is basically kinematics chain of rigid links interconnected by movable joints. The hand is also called end effectors. The end effectors may be a tool or gripper or any other device to do the work. The end effector is similar to the human hand with or without fingers.

3.8.2 Purpose

The end effectors, or robotic hand, can be designed to perform any desired task such as welding, gripping, spinning etc., depending on the application. For example, robot arms in automotive assembly lines perform a variety of tasks such as welding and parts rotation and placement during assembly. In some circumstances, close emulation of the human hand is desired, as in robots designed to conduct bomb disarmament and disposal**.**

3.9 DC Motor

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

DC motors were the first type widely used, since they could be powered from

existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications, [28].

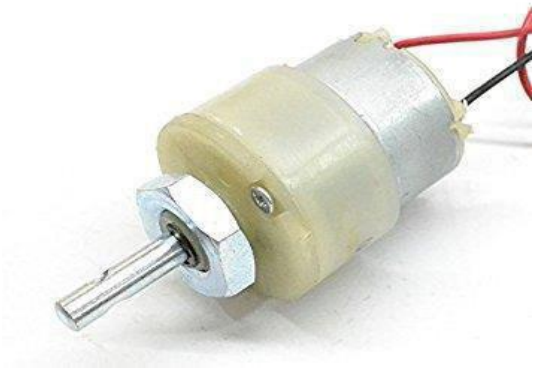

Fig. 3.30: Dc Gear Motor

3.9.1 Features of Dc Motor

- 100RPM 12V DC motors with Gearbox
- 3000RPM base motor
- 6mm shaft diameter with internal hole
- 125gmweight
- Same size motor available in various rpm
- 1.2kgcmtorque
- No-load current = 60 mA (Max), Load current = 300 mA (Max)

3.9.2 Interfacing dc motor with Arduino UNO

The dc motors are controlled by interfacing it with Arduino Uno. This dc motors are used to rotate the wheels of robot. Gear motor is a specially designed DC motor whose gear assembly helps in increasing the torque and reducing the speed. Compared to a normal DC motor, maximum rpm a gear motor can produce is less. But they have the advantage that by

using the correct combination of gears, its rpm can be reduced to any desirable value. Unlike servo motor, gear motor can also be rotated continuously**.** The direction of the gear motor can be reversed by simply reversing the polarity of the battery connection. And the speed of the motor can be controlled by changing the voltage level across it, [29,30].

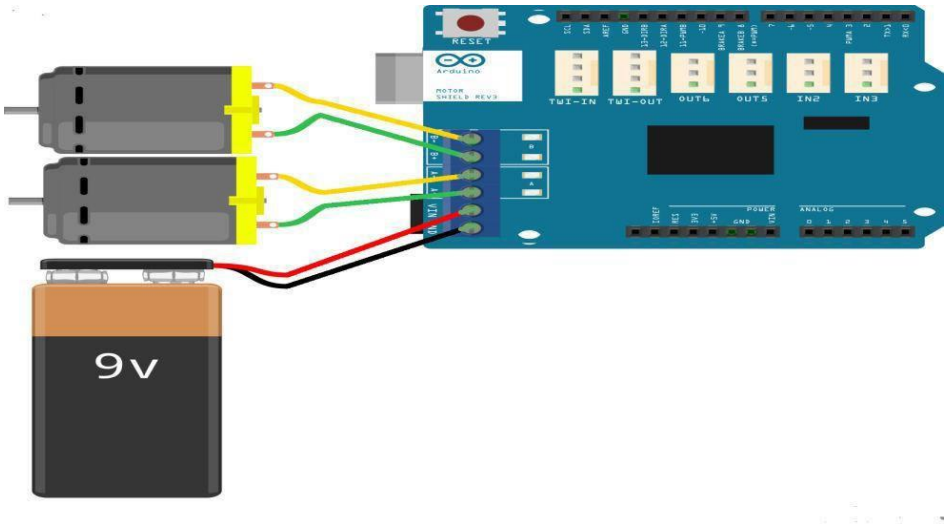

Fig. 3.31: Interfacing Schematic Diagram for Dc Motors and Arduino UNO

The dc motor connections are as shown below

- IN1 of dc motor is connected to 5V pin of Arduino UNO.
- IN2 of dc motor is connected to GND pin of Arduino UNO.

Table. 3.7: Motor Directions for Different Input Conditions

Input Pin 1	**Input Pin 2**	**Motor Direction**
Low	High	Turn Right
High	Low	Turn Left
High	High	Stop
Low	Low	Stop

3.10 External Power Battery

A nickel–metal hydride battery, abbreviated NiMH or Ni–MH, is a type of rechargeable battery, [31]. The rechargeable battery, storage battery, secondary cell, or accumulator is a type of electrical battery which can be charged, discharged into a load, and recharged many times, as opposed to a disposable or primary battery, which is

supplied fully charged and discarded after use, [32].

It is composed of one or more electrochemical cells. The term "accumulator" is used as it accumulates and stores energy through a reversible electrochemical reaction. Rechargeable batteries are produced in many different shapes and sizes, ranging from but to cells to megawatt systems connected to stabilize an electrical distribution network. Several different combinations of electrode materials and electrolytes are used including lead–acid, nickel–cadmium (NiCd), nickel–metal hydride (NiMH), lithium-ion (Li-ion)and lithium- ion polymer (Li-ion polymer).

Fig. 3.32: Power Battery

The nine-volt battery, or 9-volt battery, is a common size of battery that was introduced for the early transistor radios. It has a rectangular prism shape with rounded edges and a polarized snap connector at the top. This type is commonly used in walkie-talkies, clocks and smoke detectors, [33].

3.11 Tools Used

Software

- Arduino IDE

Hardware

- Arduino UNO
- Color Detection Sensor(TCS3200)
- Servo Motors
- Dc Motors
- Robotic Arm
- External Power Battery

4.1 The Harvesting robot setup

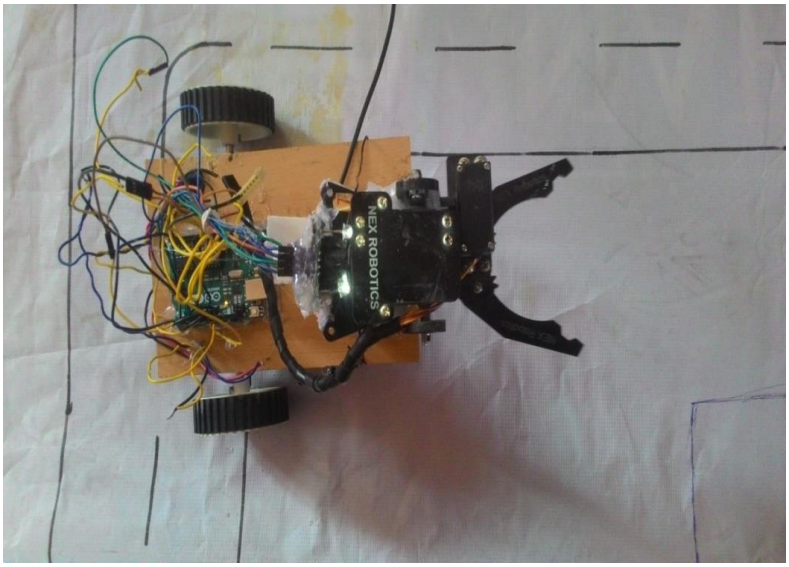

Fig. 4.1: Harvesting robot setup

The above fig 4.1 shows the experimental setup for harvesting robot. The above setup is made by interfacing color detection sensor, robotic arm with Arduino UNO. The entire harvesting robot is controlled by Arduino UNO.

4.2 Robot moving towards the tree

Fig. 4.2: Forward movement of the harvesting robot

The robot is moving forward when it is powered on. The above Fig. 4.2 shows that the robot moving towards the tree when it senses the Red color. The robot moves to forward until it reaches to the tree.

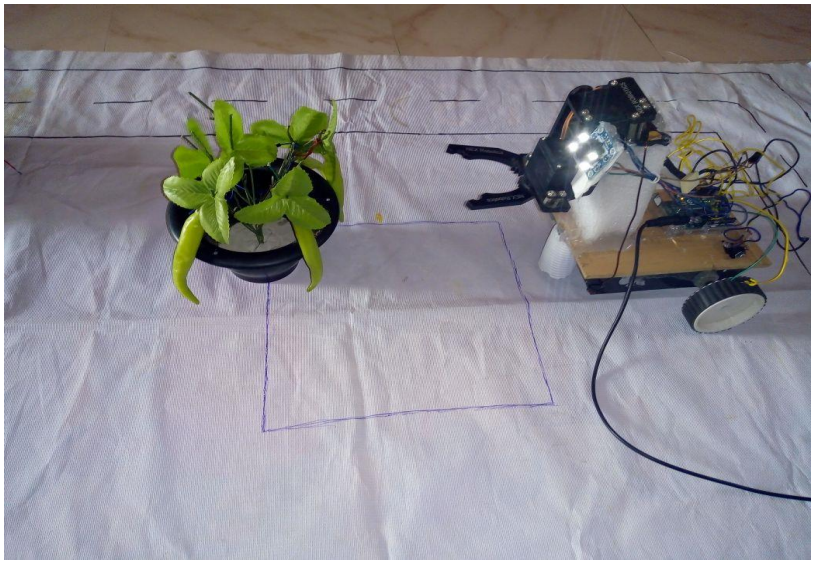

Fig. 4.3: side view of the robot while moving forward

. The above Fig. 4.3 shows that the robot moving towards the tree when it senses the green color. The robot moves to forward until it reaches to the tree.

4.3 Robot plucking the fruit

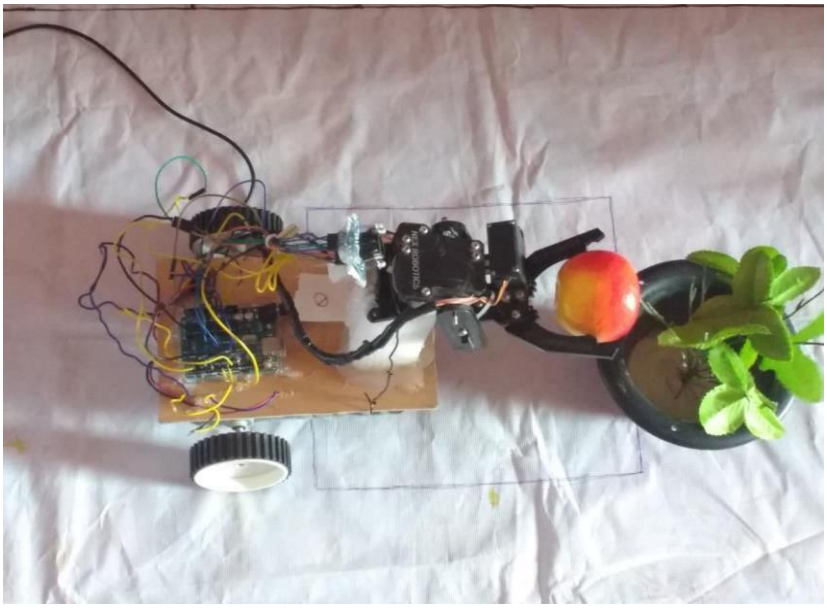

Fig. 4.3: Gripping action of the robot (Apple)

The above prototype shows the harvesting robot while plucking the fruit. In this fig the robot plucks fruit from the tree when it detects the red colored fruit. In this the Arduino is programmed for color detection sensor to detect the fruit location based upon the color intensity of the fruit.

4.4 Robot plucking the chillies

Fig. 4.4: Gripping action of the robot (chillies)

The above Fig. 4.4 shows the harvesting robot while plucking the fruit. In this the robot plucks chillies from the tree when it detects the green colored chilly. In this the Arduino is programmed for color detection sensor to detect the vegetable location based upon the color intensity value.

5.1 Conclusion

The harvesting robot which is being designed by us once comes in to use, it will solve lot of problems faced by fruit growers during harvesting. The problems like shortage of skilled labor long process of manually plucking of fruits, damage of fruits, over ripened of fruits, high cost of labor and danger for the labor personnel from thorns and reptiles in the gardens etc. are the major hurdle for the fruit growers. The harvesting robot is ultimate solution to the above said problems and it will eliminate the hurdle of fruit growers. As a result the fruit growers will be encouraged to grow more and more fruits. Also the use of harvesting robot is an important step in the modernization of technology in the field of agriculture.

5.2 Future Scope

The proposed harvesting robot is limited in detecting multi colored fruits. As the harvesting robot able to detect only primary colored fruits, for multi colored fruit detection and harvesting go with image/video processing technology. By replacing color detection sensor with Digital camera we can improve the advancement for this project using image or video processing technology.

References

1. 8051 Microcontroller-Internals, Instructions, Programming & Interfacing by Subrata Ghoshal.
2. 8051 Microcontroller based Embedded Systems by Manish K
3. Getting started with Arduino by Massimo Banzi
4. Beginning Arduino by Michael McRoberts
5. Arduino: 101Beginners Guide by Erik Savas
6. https://en.wikibooks.org/wiki/Robotics
7. https://www.hgtv.com/design/outdoor-design/landscaping-andhardscaping/tips-and-tools- for-harvesting-fruit
8. https://www.google.co.in/search?client=msandriodomlge&biw=360&bih=513&q=twist+a nd +pull+method+for+fruits&oq=tw&aqs=mobile-gws-lite.0.35i39l1jol4
9. https://forums.ni.com/t5/Projects-Products/Fruit-Plucking-Robot-for-Student- Design-Competition-2013/ta-p/3516727
10. https://en.wikipedia.org/wiki/Arduino_Uno.
11. https://en.wikipedia.org/wiki/ATmega328.
12. https://www.google.co.in/search?ei=jiaqWs2MIcnsvgTxvrb4Cw&q=atmega328+microcontroller+REFERENCES&oq=atmega328+microcontroller+REFERENCES&gs_l=psyab.3...5944.12278.0.13148.11.11.0.0.0.0.501.2015.0j4j2j1j0j1.8.0....0...1.1.64.psyab..3.4.1192...0i13k1j33i160k1.0.qrMu2Q8a15c
13. https://startingelectronics.org/software/arduino/installing-arduino-software-windows 7/
14. http://www.instructables.com/id/Intro-to-Arduino/
15. https://www.arduino.cc/en/Reference/Board
16. https://www.arduino.cc/reference/en/language/structure/control-structure/while/
17. www.dfrobot.com/wiki/index.php/TCS3200_color_sensor_(SKU:SEN010)
18. https://robokits.co.in/sensors/color-sensor/tcs3200-based-color-sensor-module
19. https://create.arduino.cc/projecthub/mjrobot/arduino-color-detection-57e4ce
20. https://www.google.co.in/search?q=color+detection+sensor+arduino&sa=X&ved=0

ahU KEwjel6-3ZAhUCTY8KHRahDI0Q1QIIhQIoAA&biw=1280&bih=615

21. https://www.jameco.com/jameco/workshope/howitworks/how-servo-motors-work.html

22. https://www.theengineeringprojects.com/2017/05/servo-motor-contro using-arduino.html

23. http://microcontrollerslab.com/servo-motor-control-and-interfacing-with-arduino/

24. https://circuitdigest.com/microcontroller-projects/arduino-servo-motor-control-code.

25. https://en.wikipedia.org/wiki/Robotic_arm.

26.https://www.google.co.in/search?q=robotic+arm+introduction&source=lnms&tbm =isch&sa=X&ved=0ahUKEwjw8oyKxe7ZAhUW448KHcrHCB8Q_AUICygC& biw=1350&bih= 569

27.https://www.google.co.in/search?biw=1350&bih=569&tbm=isch&sa=1&ei=mIOq Ws_1OYrrvATg6ai4Cw&q=robotic+arm+&oq=robotic+arm+&gs_l=psyab.3..0l9j 0i67k1.61120.63258.0.64643.12.8.0.0.0.0.392.952.0j4j0j1.5.0....0...1c.1.64.psyab.. 7.5.950...0i24k1j0i30k1.0.7Twgo5kifSQ

28. https://circuitdigest.com/microcontroller-projects/dc-motor-control-with-arduino-uno- pwm

29. http://www.electronicwings.com/arduino/dc-motor-interfacing-with-arduino-uno

30. https://elementztechblog.wordpress.com/2017/12/07/interfacing-dc-motor-with-arduino/

31. https://en.wikipedia.org/wiki/Nickel%E2%80%93metal_hydride_battery

32. https://en.wikipedia.org/wiki/Rechargeable_battery

33. https://en.wikipedia.org/wiki/Nine-volt_battery

34. https://arduino.stackexchange.com/questions/816/c-vs-the-arduino-language

35. https://www.quora.com/What-programming-language-is-used-to-program-an-arduino-board

36.https://www.informationvine.com/index?qsrc=999&qo=semQuery&ad=semD&o= 60302&l=sem&askid=2f08ecab702b4e12a9448d16698b40b90iv_gsb&q=arduino %20uno%20pro gramming%20codes&dqi=&am=broad&an=google_s

37.https://www.zapmeta.ws/ws?q=arduino%20programming%20code&asid=ws_gc19_

05&mt=b&nw=g&de=c&ap=1o3

38. https://www.makerspaces.com/arduino-uno-tutorial-beginners

39.https://www.informationvine.com/index?qsrc=999&qo=semQuery&ad=semD&o=603902&l=sem&askid=2f08ecab702b4e12a9448d16698b40b90iv_gsb&q=arduino%20uno%20pr ogramming%20codes&dqi=&am=broad&an=google_s

CODE FOR HARVESTING FRUITS

```
// TCS230 or TCS3200 pins
wiring to  Arduino

 #define S0 4

#define S1 5

#define S2 6

#define S3 7

#define sensorOut 8

#define s4 12

#define s5 10

#define s6 2

#define s7 9

 #include<Servo.h>

Servo servo1;

Servoservo2;

Int pos1=0;

int pos2=0;

// Stores frequency read by the

photodiodes

Int redFrequency = 0;

Int greenFrequency = 0;

 Int blueFrequency = 0;

 void setup() {
```

```
// Setting the outputs
servo1.attach(9);
servo2.attach(2);
pinMode(S0,OUTPUT);
pinMode(S1,OUTPUT);
pinMode(S2,OUTPUT);
pinMode(S3,OUTPUT);
pinMode(S4,OUTPUT);
pinMode(S5,OUTPUT);
pinMode(S6,OUTPUT);
// Setting the sensorOut as an input
pinMode(sensorOut, INPUT);
// Setting frequency scaling to 20%
digitalWrite (S0, HIGH);
digitalWrite (S1, LOW);
// Begins serial communication
Serial.begin(9600);
 }
 void loop() {
// Setting RED (R) filtered photodiodes to be read
digitalWrite (S2,LOW);
digitalWrite (S3,LOW);
// Reading the output frequency
```

```
redFrequency = pulseIn(sensorOut, LOW);
 if (redFrequency< 100)
   {
digitalWrite (s4,HIGH);
digitalWrite (s5,HIGH);
    }
else if(redFrequency>100)
{
 for( pos1 = 0 ; pos1 <180; pos1 += 1)
 {servo1.write(pos1);
 delay (10);
   }
 for( pos1 = 180 ; pos1 >=1; pos1 -= 1)
   {servo1.write(pos1)
    delay (10);
    }
 for( pos2 = 0 ; pos2 <180; pos2 += 1)
 {servo2.write(pos2);
 delay (10);
   }
 for( pos2 = 180 ; pos2 >=1; pos2 -= 1)
   {servo2.write(pos2);
 delay (10);
```

```
    }
digitalWrite (s4,LOW);
digitalWrite (s5,LOW);
   }
else
digitalWrite (s4,LOW);
digitalWrite (s5,LOW);
   }
// Printing the RED (R) value
Serial.print("R = ");
Serial.print(redFrequency);
delay(100);
// Setting GREEN (G) filtered photodiodes to be
read
digitalWrite (S2,HIGH);
 digitalWrite (S3,HIGH);
// Reading the output frequency
greenFrequency = pulseIn(sensorOut, LOW);
// Printing the GREEN (G) value
Serial.print(" G = ");
Serial.print(greenFrequency);
delay(100);
```

```
// Setting BLUE (B) filtered photodiodes to be

read

digitalWrite (S2,LOW);

digitalWrite (S3,HIGH);

// Reading the output frequency

blueFrequency = pulseIn(sensorOut, LOW);

// Printing the BLUE (B)

value Serial.print(" B = ");

Serial.println(blueFrequency)

; delay(100);

  }
```