



Palestine Polytechnic University
Collage of information Technology and
Computer Engineering

Low-Cost 6DoF Input Device

Team:

Ghaeda' Jawad Murrar

Supervisor:

Dr. Zein Salah

Acknowledgement

First and foremost, we would like to thank God Almighty for giving us the ability and opportunity to undertake this project and help us to get through it. Without his blessings, this achievement would not have been possible.

Our heartiest thanks are due to our parents for their love and kind cooperation which helped us in the project. We know the thanks is not enough and there are not enough words to describe how thankful we are.

Special thanks to my husband, Alaa Etkadek, who was always my inspiration power and positive energy, his words mean the world to my heart.

We take this opportunity to express our regards and sincere thanks to our supervisor Dr. Zein Salah. His constant encouragement and moral support gave us the motivation to get through this project successfully.

Last but not least we would also like to thank our families, friends and all the teachers for their support. We are also thankful to everyone who supported us and gave us many helpful comments.

Abstract

Users of three-dimensional computer-aided design (CAD), computer simulations and gaming applications need to manipulate 3D objects in up to six degrees of rotational and translation freedom (6DoF). To date, no 3D controller provides one-handed 6DoF input with miniature size and low cost, all existing 6DoF controllers compute 3D object's position and orientation using high materials and multiple sensors that make it relatively expensive and complicated in use.

The aim of this project is to construct a low-cost 6DoF input device system, this system allows both expert and nonexpert users to intuitively control 3D objects in up to 6DoF using low-cost input device.

We addressed the issue of controlling 3D objects in 6DoF by using outside-in optical tracking technique that is based on one low-cost infrared camera connected to ATmega1284P AVR microcontroller, the camera tracks a well-designed pattern of four active markers mounted at the front of an input controller.

On the microcontroller, 3D pose estimation algorithms would be applied on the positional information of the tracked markers that comes from the camera. a device driver was developed to receive the calculated pose data of the input controller and apply it to the controlled 3D object in 3D application. The system communicates with a 3D application via Serial/USB connection between the microcontroller and the device driver hosted on the PC.

Table of Content

ACKNOWLEDGEMENT	II
ABSTRACT	III
LIST OF TABLES	VI
LIST OF FIGURES	VII
INTRODUCTION	1
1.1.Overview of the project.....	1
1.2.Motivation and Importance.....	1
1.3.Objectives	2
1.4.Description of the Project.....	2
1.5.Problem Analysis.....	3
1.6.List of Requirements	3
1.7.Expected Results	3
1.8.Project Time Line	4
1.9.Project Gantt Chart	5
1.10.Report Outline	5
LITERATURE REVIEW AND BACKGROUND	6
2.1.Literature Review.....	6
2.1.1.6DoF input devices	6
2.1.2.Comparison between 6DoF controllers and our project.....	8
2.2.Background	9
2.2.1.Theoretical Background.....	9
2.2.2.System Hardware Components	11
2.2.3.System Software Components	16
SYSTEM DESIGN	17
3.1. Design options	17
3.2.Flowcharts.....	20
3.2.1.System Flowchart.....	20
3.2.2.Device Driver Flowchart.....	21

3.3.Diagrams	23
3.3.1.Block Diagram	23
3.3.2.Wiring Diagrams	24
3.3.3.Schematic Diagrams.....	27
HARDWARE AND SOFTWARE IMPLEMENTATION	29
4.1Overview	29
4.2Hardware Implementation	29
4.2.1Receiving side Implementation:	29
4.2.2Transmitting side Implementation:	30
4.3 Software Implementation.....	31
4.4 Implementation Issues.....	34
4.5 Implementation Results.....	35
VALIDATION AND TESTING	36
5.1Overview	36
5.2Hardware Testing.....	36
5.2.1Testing Atmega1284P MCU:.....	36
5.2.2 Testing the FTDI chip with the Atmega1284P MCU:	37
5.2.3 Testing the Pixy Camera:	37
5.3Software Testing	37
5.3.1 Testing the core code:	37
5.3.2 Testing the device driver:	39
5.3.3 Testing the 3D graphical application:	39
5.4System Testing.....	39
CONCLUSION	40
6.1Summary	40
6.2Challenged faced.....	40
6.3Future works	40
References	41

List of Tables

Table 1: Project Time Line	4
Table 2: Project Gantt Chart	5
Table 3: Comparison between existing 6DoF controllers and our project	8
Table 4: Requirement Analysis.....	17
Table 5: Pixy Camera Pin Map with ATmega1284P	24
Table 6: FTDI Serial/USB pin map with ATmega1284P	25
Table 7: Pixy Camera Pin Map with ATmega1284P	26
Table 8: Marker Detection Testing Results	37
Table 9: 6DoF Pose Detection Testing	38

List of Figures

Figure 1: Space mouse pro.....	6
Figure 2: Space mouse wireless	7
Figure 3: Spaceball 5000	7
Figure 4: Daydream 6DoF controllers	8
Figure 5: Six degrees of freedom.....	10
Figure 6: Top view of the pattern of four IR markers.....	11
Figure 7: ATmega1284P AVR Microcontroller.....	11
Figure 8 : Wii IR Camera.....	12
Figure 9: Pixy Camera	13
Figure 10: LEDs	14
Figure 11: Air Presenter Remote Mouse	14
Figure 12: FTDI Serial/USB Interface.....	15
Figure 13: System Flowchart	20
Figure 14: Device Driver Algorithm Flowchart	22
Figure 15: System Block Diagram.....	23
Figure 16: Connecting Atmega1284P with Arduino Mega for boot loading	24
Figure 17: Interfacing FTDI Serial/USB to ATmega1284P.....	25
Figure 18: Pixy Camera to Atmega1284P	26
Figure 19: Whole System design	26
Figure 20: Boot loading ATmega1284P using Arduino Mega	27
Figure 21: Schematic diagram of FTDI Serial/USB to ATmega1284P.....	27
Figure 22: Schematic diagram of Pixy Camera to ATmega1284P	28
Figure 23: Schematic diagram of the whole system	28
Figure 24: Implementation of Atmega1284P Bootloading.....	29
Figure 25: All components together	30
Figure 26: Brightness Problem	30
Figure 27:Front side of the pointer	31
Figure 28: GUI of the 3D Graphical application	34
Figure 29: 6DoF Input device.....	35
Figure 30: Testing the bootloader	36
Figure 31: Test FTDI	37

Chapter 1

Introduction

1.1. Overview of the project

Of the several factors guiding the global computer graphics market towards a brighter future, the leading driver today is the growing use of image processing and 3D animation effects in the media, computer-aided design (CAD) and entertainment industry.

The growing use of 3D animation is prompting a heavier competition between hardware manufacturers in order to reduce product costs and increase product quality.

To create intuitive 3D-interaction between 3D graphical applications and human, using 3D representations on displays is not enough. Users need to manipulate 3D applications in a more natural way. To achieve that, 3D-interaction input devices are required, such as 3D mouse.

However, many existing 3D mouse devices are either often too expensive to buy by majority of users or don't satisfy the need of intuitive 3D-interaction and don't give a full 6DoF controlling. So, with our little materials, we came with our project, designing a low-cost 6DoF input device system.

1.2. Motivation and Importance

In looking at the need of low cost and intuitive 6DoF input device for object manipulation and navigation in many 3D applications like CAD programs, virtual reality games and computer simulations, the existing 6DoF input devices compute 3D object's position and orientation using high materials and multiple sensors that make it often expensive, besides high processing costs and sophisticated analysis that make it unintuitive devices.

The most important motivation for this project is to overcome this problem, the 3D mouse developed in this project, is a low-cost 6DoF input device that can be used in 3D-interactive applications.

This achieves several points; the most important point is to provide a simple 6DoF input device that allows the inexperienced users to intuitively manipulate 3D-interactive applications without having to know much about 3D object manipulation, in addition to providing a low-cost 6DoF input device to capture some share of market.

1.3. Objectives

The main objective of this project is to design a low-cost 6DoF input device system that provides intuitive moving and rotating of 3D objects in the 3D graphical applications. In essence, we aim to design and implement a system that:

1. enables the users to control 3D objects in 6DoF movements.
2. is easy and intuitive, as the user moves our handheld device, the controlled object directly mimics the movement without perceivable latency.
3. is low-cost to capture some share of market.
4. accurately detects the object's pose through calculating its 3D position (X, Y, Z) and orientations (Yaw, Pitch, Roll) at high frequency and low processing cost.

1.4. Description of the Project

This project is mainly designed to help get over the problem of intuitive 3D-interaction with 3D graphical applications. We will design a low-cost 6DoF input device, which allows the user to get the feeling of holding the object in his hand.

In details, the system has two parts, at the transmitting part there is a handheld input device with pattern of four IR markers. As the user moves the input device, an IR camera sensor located in the receiving part tracks the four IR markers, and then gives out the pixel position for each IR marker to the microcontroller.

Then the microcontroller calculates the 6DoF pose of the input device at high frequency through applying 3D pose estimation algorithms to calculate the 3D position (X, Y, Z) and orientations (Yaw, Pitch, Roll) of the input device, then sends the result to the device driver hosted on a PC to show the user movement and rotation on the controlled 3D object.

1.5. Problem Analysis

At the present, 6DoF interaction of input devices is becoming one of the most important research fields that has to be enhanced to reach the sufficient level of 3D-interaction between 3D graphical applications and users.

The design of the low-cost 6DoF input device system, presented in this project, would be an added value for 3D-interaction field due to its simplicity, low-cost, high performance and accuracy in calculating the 6DoF pose of the input device.

1.6. List of Requirements

In this section, we present the main requirements of our system:

1. 6DoF Control: The system would free up the range of interactions between user and 3D application. It allows the user to instantaneously, precisely and reliably control 3D objects.
2. Low-cost: The used components have to be available at low prices, so the user would have a high-quality 6DoF mouse in low price relative to the existing 6DoF input devices.
3. Intuitive: The user would move and rotate 3D objects exactly as if it is held in a hand, because the system would guarantee robustness so that small motions cause small changes in the controlled object.
4. Handy: The input device in the system should be wireless and handheld.
5. Resource saving: The system would be based on one sensor that tracks user movements, no multiple sensors, no data link and with minimalist processing costs.
6. Versatile: The system should be compatible to work with PCs.

1.7. Expected Results

We expect to accomplish the following at the end of the project:

1. Build a device driver to interface the system to a 3D application on PC.

2. Low-cost 3D input device system will be built in final form to accomplish the purpose defined in the previous sections.
3. Intuitively move and rotate the controlled 3D object in response to the movement of the handheld input device.

1.8. Project Time Line

Table 1: Project Time Line

Task No.	Task Name	Duration (Weeks)
1.	Project planning	4
2.	Determination project requirements	4
3.	Project design and analyzing	8
4.	Project development	10
5.	Project testing and maintenance	4
6.	Documentation	28

1.9. Project Gantt Chart

Table 2: Project Gantt Chart

Task	Duration (Weeks)													
	First Semester							Second Semester						
	2	4	6	8	10	12	14	2	4	6	8	10	12	14
Planning	■	■												
Project Requirements			■	■										
Analyzing and design				■	■	■	■							
Project development								■	■	■	■	■		
Project testing and maintenance										■	■	■	■	■
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■

1.10. Report Outline

This report is organized as follows: Chapter 2 introduces some literature review including available 6DoF input devices and related projects. It also goes briefly over the theoretical background of the project, hardware and software components. Chapter 3 discusses the conceptual design of the system, block diagrams, flowcharts and detailed hardware connections. Chapter 4 discusses the hardware and software implementations of the project. Chapter 5 shows the testing process which is made until we reach the final system design. Finally, Chapter 6 presents the final results of the project, future works and conclusion.

Chapter 2

Literature Review and Background

2.1. Literature Review

There are several variations of 3D mouse devices with 6DoF controlling of objects. These types of input devices appear with many features. In the next two sections, we present a discussion about existing 6DoF input devices, in addition to a comparison between them.

2.1.1. 6DoF input devices

This section will show the famous 6DoF input devices in the markets, their specifications, features and limitations.

A. 3Dconnexion Space Mouse Pro

Description

Space Mouse Pro is a professional 6DoF controller shown in figure [1] designed for manipulating objects in a 3D environment. It features the patented 3Dconnexion 6DoF sensor that enables you to easily manipulate digital models in 3D space. The user needs to simply push, pull, twist or tilt the cap a fraction of inch to simultaneously pan, zoom and rotate 3D imagery precisely and intuitively [1].

Features

6DoF navigation of 3D models in 3D environments, supplementary function keys that can be programmed by the user for quick access and shortcuts for modeling a 3D object, display screen.

Limitations

Relatively expensive, stationary 3D controller, more suitable for expert designers than ordinary users, needs time to deal with it.



Figure 1: Space mouse pro

B. 3Dconnexion Space Mouse Wireless

Description

Space Mouse Wireless shown in figure [2] allows users to manipulate 3D models due to its unique 6DoF technology using patented 3Dconnexion 6DoF sensor. It is developed specifically to enhance user experience with intuitive, effortless and precise 3D navigation in CAD applications by simply push, pull, twist or tilt the 3Dconnexion controller cap [2].



Figure 2: Space mouse wireless

Features

6DoF navigation for 3D models in CAD applications.

Limitations

Relatively expensive, stationary 3D controller, designed to use in CAD programs, otherwise, it won't work well with other 3D-interactive applications like games.

C. 3DConnexion Spaceball 5000

Description

Spaceball shown in figure [3] allows the user to control and navigate 3D scene, by operating a sphere, which is placed on the controller that contains a 6DoF optical motion sensor. This mouse should operate together with another input device, so the user should simultaneously pan, zoom or rotate 3D models with the Spaceball in one hand, while the other hand selects, inspects or edits with, for example, a traditional mouse [3].



Figure 3: Spaceball 5000

Features

It is two-handed 6DoF input controller, fits most of 3D applications, has programmable buttons.

Limitations

It is designed to operate with other input device like a keyboard or a traditional mouse so it wouldn't work well alone, stationary controller, needs time to learn how to deal with it.

D. Daydream 6DoF controllers - experimental

Description

These controllers shown in figure [4], which are only available for developers at this time, feature 6DoF controlling using 6DoF optical tracking ball, they bring natural and intuitive controlling to the virtual reality headset, Lenovo Mirage Solo VR headset, specifically [4].

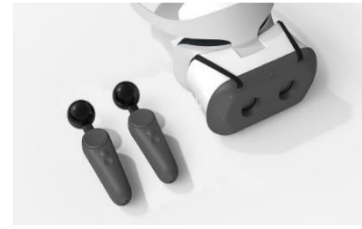


Figure 4: Daydream 6DoF controllers

Features

It provides 6DoF controlling to VR headsets, handheld device.

Limitations

It is specified to work with VR Lenovo Mirage Solo headset only.

2.1.2. Comparison between 6DoF controllers and our project.

This section shows a comparison between most common 6DoF controllers in the markets which are mentioned in the previous section with our project.

Table 3: Comparison between existing 6DoF controllers and our project

Comparison	Cost/Price	Size	Could held in the hand	Easy to use by ordinary users	Uses multiple sensors	Works independently
3Dconnexion Space Mouse Pro	300.00 \$ - 500.00 \$	Average	No (Stationary)	Bad	Yes	Yes
3Dconnexion Space Mouse Wireless	150.00 \$ - 200.00 \$	Average	No (Stationary)	Good	Yes	Yes
3DConnexion Spaceball 5000	100.00 \$ - 200.00 \$	Average	No (Stationary)	Bad	Yes	No

Daydream controllers- experimental	6DoF	Cost is not specified	Small and light	Yes (Handheld)	Good	Yes	No
This Project		< 40 \$	Small and light	Yes (Handheld)	Good	No	Yes

2.2. Background

This section briefly describes the theoretical background of the project, and shows short description of the hardware and software parts which are used in the system.

2.2.1. Theoretical Background

This section provides information about some used technologies and techniques in the project.

Outside-in Optical Tracking Technique

Marker-based Outside-in Tracking is a method of optical tracking. In this method a target is fitted with markers which form a known pattern. A camera or multiple cameras installed at known location in the environment constantly seek the markers then use various algorithms to extract the position of the object from the markers. By synchronizing the time that the markers emit with the camera, it's easier to block out other IR lights in the tracking camera, so the camera detects the energy emitted from the active markers and translates it to positional information using various techniques like, for example, triangulation ^[5].

In our project, we intend to use this technique of tracking because of its simplicity and robustness in tracking, one camera will be installed at the PC side, a pattern of four active markers will be mounted at the front of the input controller, simply that allows the camera to tracks the pose of the controller (target) by tracking the markers installed on it.

Six Degrees of Freedom (6DoF) Controlling

Degrees of freedom (DoF) refer to the number of basic ways an object can move through 3D space. There are six total degrees of freedom. 3DoF correspond to rotational movement around

the x, y, and z axes, that are commonly named pitch, yaw, and roll respectively. The other three degrees correspond to translational movement along those axes, which can be thought of as moving forward or backward, moving left or right, and moving up or down, figure 5 represents the 6DoF of an object.

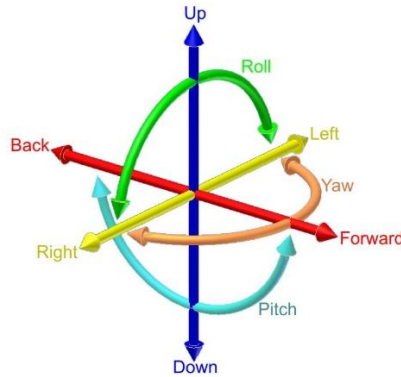


Figure 5: Six degrees of freedom

Pattern of Non-Colinear Markers Technique

This Technique is used to solve many object pose detection and localization problems by tracking these objects with specific designed pattern of markers that is tracked from one or more cameras to applying mathematical calculations on the movement of these markers and then output the object pose.

In our project, mathematical processing using this technique will be used to determine the 3D coordinates and angular orientation i.e. 6DoF of the controller. These calculations applied to the lights emitted from the pattern markers, so the design of the pattern of four markers plays an important role in system precision. The aim of using four markers not 1 or 2 or 3 is explained in the following points:

- 1- Using just 1 marker means that we can just detect a point (X, Y) in the 3D space without any more useful information about the position and the orientation of this object.
- 2- Using 2 markers means that we can detect (X, Y, Z), but sure these two points are on the same line so this will make collusion.

- Using 3 non-collinear markers is enough for 6DoF pose estimation but it won't avoid ambiguity.

So, at the end, figure [6] below shows approximate final top view of the pattern design.

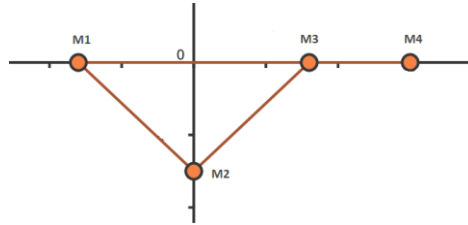


Figure 6: Top view of the pattern of four IR markers

2.2.2. System Hardware Components

This section shows the hardware components which are used in this project, and how they are used.

1. ATmega1284P AVR Microcontroller

The ATmega1284P shown in figure [7] is a low-power 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega1284P achieves throughputs close to 1MIPS per 1MHz. This empowers system designer to optimize the device for power consumption versus processing speed [6].

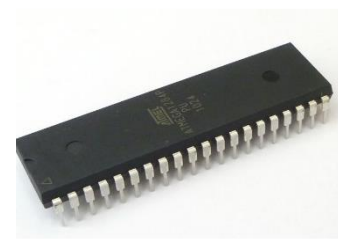


Figure 7: ATmega1284P AVR Microcontroller

Specifications

The specifications that are interesting for this project are listed below [8]:

- Clock speed: 16 MHz.
- RAM: 128 KByte.
- Hardware support for I2C.
- Its port C has (SDA, SCL) pins that are Serial Bus Data Input/Output Line and Serial Bus Clock Line respectively, to connect the camera.
- Its port D has (TXD0, RXD0) pins that are transmitter and receiver lines respectively, to connect with the Serial/USB interface to the PC.

Usage

The ATmega1284P is the controller. It reads the camera data, does all the calculations and outputs the 6DoF pose of the controlled object i.e. its 3D coordinates (X, Y, Z) and 3D orientations (Yaw, Pitch, Roll), then sends it to the device driver on the PC via Serial/USB connection. It's pretty suitable for our project because it's small, light and cheap enough to use.

2. Wii IR Camera

The Wii Remote shown in figure [8] +-includes a 128x96 monochrome camera with built-in image processing. The camera looks through an infrared pass filter in the remote's plastic casing. The camera's built-in image processor is capable of tracking up to 4 moving objects. The built-in processor uses 8x subpixel analysis to provide 1024x768 resolution for the tracked points ^[7].

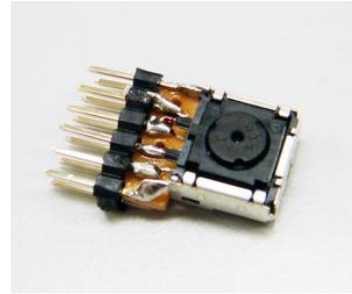


Figure 8 : Wii IR Camera

Specification

The specifications that are interesting for this project are listed below ^[7]:

- Hardware Interface: I2C
- Built-in image processor.
- Resolution: 1024x768 pixel for the tracked objects.
- Field of View: 33 degrees horizontally, 23 degrees vertically
- IR-Pass Filter: 940nm on to lenses as well as plastic filter in front of the camera.

Usage

We need IR camera to track the pattern of four IR markers mounted at the front of the input device, the camera shown in figure 8 is a Wii IR camera that is disassembled from Wii Remote as standalone sensor offers a low-cost solution for absolute optical tracking due to its ability to track up to four markers simultaneously. Since its tracking is hardware-based, it's fast. Moreover, it gives out the coordinates of each tracked marker via an I2C bus connected to a microcontroller.

.....

But, Unfortunately, we faced a lot of limitations and challenges through using this camera, we collect 5 Wii IR Cameras from different used and new Wii Remotes, the problem we faced with each of these cameras is as follows:

- Camera 1: It has 8 pins as the original Wii IR Camera, but the arrangement of the pins is slightly different from the original, also two of its pins are labeled as [L-, L+], these differences make it difficult to connect it to the Atmega1284P and taking results.
- Camera 2: It has 8 pins as the original Wii IR Camera with the same arrangement, but sadly the SDA pin is burned somehow.
- Camera 3: same as Camera 1.
- Camera 4: It has unlabeled 12 pins, we searched a lot on the internet about any guideline resources for connecting these unlabeled pins to Atmega1284P, but with no useful results.
- Camera 5: It seems like the original Camera with the same labeled pins, but when we tried to connect it to the ATmega1284P the circuit turned off, after many testing attempts, we discovered that pin SDL is burned.

After these challenges, we decided to use Pixy Camera instead of Wii IR Camera.

3. Pixy Camera

Pixy is a fast vision sensor for DIY robotics and similar applications. The User can teach Pixy an object just by pressing a button. It's capable of tracking hundreds of objects simultaneously and only provides the data you are interested in.

It doesn't return an image, it has a processor that makes processing on the camera image and then it returns data describes the object it's tracked.



Figure 9: Pixy Camera

Specifications

The specifications that are interesting for this project are listed below ^[8]:

- Processor: NXP LPC4330, 204 MHz, dual core.
- Lens field-of-view: 75 degrees horizontal, 47 degrees vertical.
- Small, fast, easy-to-use, low-cost, readily-available vision system

Usage

After turning from using Wii IR Camera as mentioned previously, we need a camera to track the pattern of four active markers mounted at the front of the input device, The camera shown in figure 9 is a Pixy camera that is used as alternative solution for absolute optical. Since its tracking is hardware-based, it's fast. Moreover, it gives out the coordinates of each tracked marker via an I2C bus connected to a microcontroller.

4. Light Emitting Diode (LED)

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

Usage

Our system performs outside-in optical tracking using LEDs as figure 10 shows, so we need to design a pattern of four active markers.



Figure 10: LEDs

5. Air Presenter Remote Mouse

This wireless remote, shown in figure 11, gives the user free movement in the room while presenting



Figure 11: Air Presenter Remote Mouse

Specification

The specifications that are interesting for this project are listed below:

- Plug-and-Play wireless receiver.
- No software to install.

Usage

We need a handheld input device that holds at the front the pattern of four LEDs markers that would be tracked from the LEDs camera, this device buttons would be remapped to customize their functionality to be as we need.

6. FTDI Serial/USB FT232RL Interface

The FT232RL chip is one of the most commonly ICs that is used to convert USB signals to serial UART signals. It allows the user to communicate with and upload code to an arduino or other microcontroller from PC. USB to serial designs using the FT232R have been further simplified by fully integrating the external EEPROM, clock circuit and USB resistors onto the device ^[12].



Figure 12: FTDI Serial/USB Interface

Specifications

The specifications that are interesting for this project are listed below ^[9]:

- Single chip USB to asynchronous serial data transfer interface.
- Data transfer rates from 300 baud to 3 Mbaud at TTL levels.
- RXD / TXD transceiver communication indicator.

Usage

This interface offers a low-cost way to add USB capability to our project, via this interface the data would be transmitted to the device driver in the PC to reflects the user movements on the controlled object.

2.2.3. System Software Components

- **Software Tools**

A. Microsoft Visual Studio (IDE)

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as drivers. Using VisualDDK that is a Visual Studio extension that allows developing and debugging device drivers, it allows debugging drivers directly from Visual Studio just like the user-mode applications. In our project, we will use it to build our device driver, it will be used to build the device driver that will be explained in section 3.2.2. ^[10].

B. Arduino 1.0.6 Software IDE

The Arduino integrated development environment (IDE) is a cross-platform application. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards like Atmega1284P that we used.

C. PixyMon Software

PixyMon is the configuration utility for Pixy Camera that runs on Windows, MacOS and Linux. In our project we use this software for monitoring the camera behavior under the testing phase.

- **Used Programming Languages**

- C++

Chapter 3

System Design

This chapter discusses the conceptual design of the system, it shows the block diagram of the system, flow charts, wiring diagrams and schematic diagrams.

3.1. Design options

Design options based on requirements analysis:

Table 4: Requirement Analysis




	Raspberry Pi Zero 	Arduino Mega R3 	ATMega1284P 	Analysis: We chose ATMega1284P microcontroller since it's available at a lower cost, also it has the required processing power that operates well with the IR camera, these specs are considered at the top of the requirements.
Requirements	Options			
1. Lower cost.	X	X	✓	
2. Fits the required processing power.	X	X	✓	
3. Suitable size.	✓	✓	✓	
4. Number of pins is sufficient.	✓	✓	✓	
5. Relatively easier in programming.	✓	✓	X	

Table 4: Continued



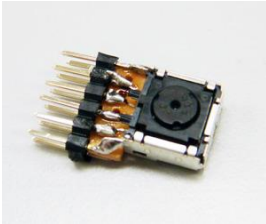



	<p>Logitech R700</p> 	<p>Air Presenter Remote Mouse</p> 	<p><u>Analysis:</u></p> <p>We chose Air Presenter Remote Mouse since it has a lower cost than the corresponding option, in addition to that, it has enough front width to mount the pattern.</p>
Requirements	Options		
1. Lower cost	X	✓	
2. Simpler.	X	✓	
3. Has enough front width to mount the IR pattern.	✓	✓	
4. Availability	✓	✓	

Table 4: Continued

	<p>Wii IR Camera</p> 	<p>Pixy Camera</p> 	<p><u>Analysis:</u></p> <p>We chose Pixy camera since it achieves all of the requirements, it is available. While Wii IR Camera has better features but it's not available.</p>
Requirements	Options		
1. Lower cost.	✓	X	
2. Suitable size.	✓	X	
3. Good resolution.	✓	✓	

4. Good field of view.	✓	✓		
5. Tracks up to four markers.	✓	✓		
6. Easy to install and to connect.	✓	✓		
7. Availability.	✗	✓		
	Atmel AVR Studio 	Arduino IDE 		<u>Analysis:</u> We chose Arduino IDE to edit and debug the code to be uploaded on the ATmega1284P, since it's open source and it's easy to use and Compatible with Atmega1284P
Requirements	Options			
1. Open source.	✓	✓		
2. Easy to use.	✗	✓		
3. Compatible with Atmega1284P	✓	✓		

3.2. Flowcharts

3.2.1. System Flowchart

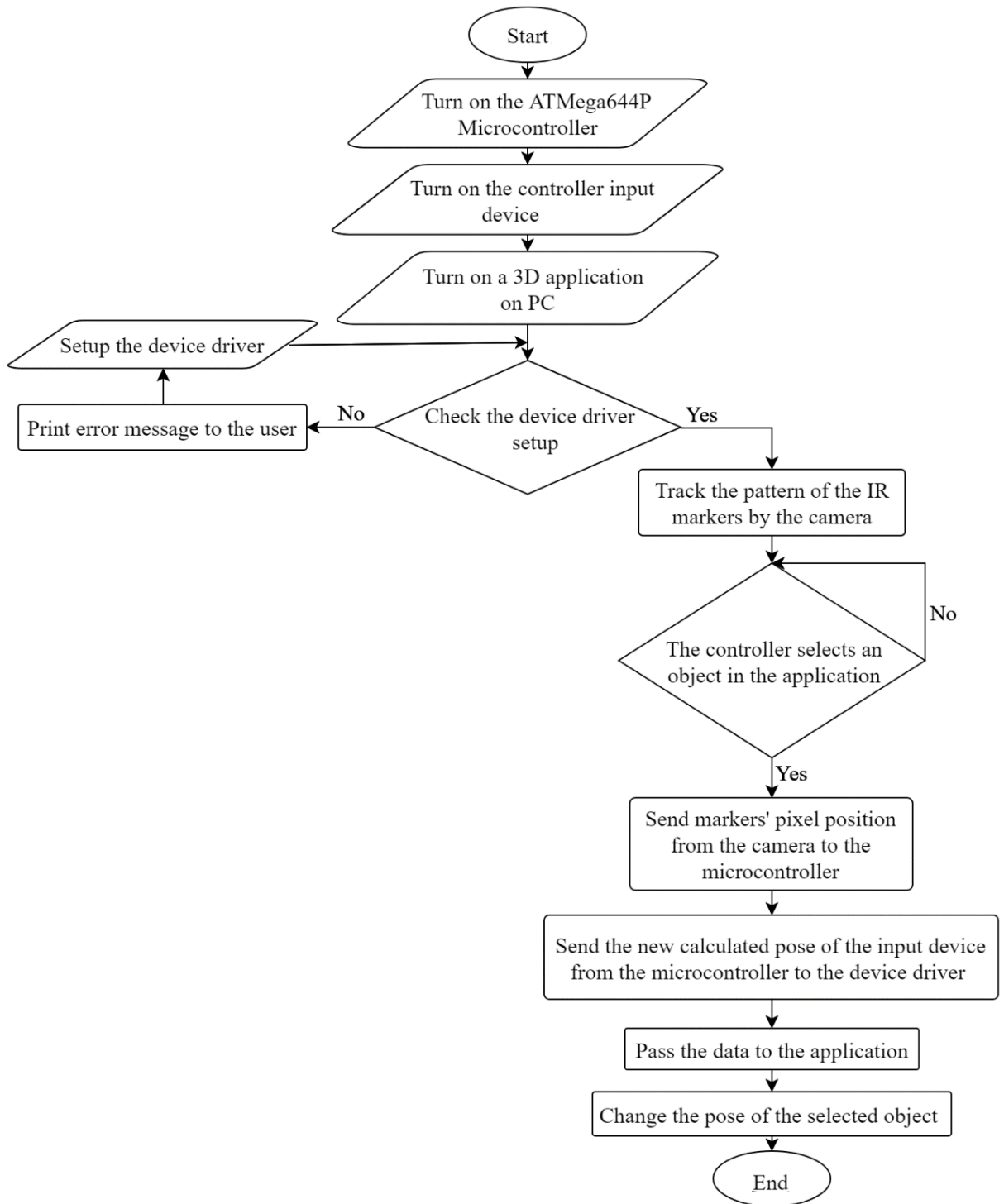


Figure 13: System Flowchart

The flowchart of the system in figure 13 shows the steps of the system, starting by activation of the ATmega1284P microcontroller and turn the controller input device on and open an 3D application on the PC, then check if the device driver is installed on the PC. If the device driver is not installed, an error message will appear to the user telling him to install the device driver on his PC. If the device driver is installed, then the tracking camera starts tracking the markers on the controller. As long the controller doesn't select any object in the application, it means that the controller doesn't control any object so its movements will not be reflected on any object in the application, once it selects an object, the microcontroller receives the position of each marker that is sent from the camera, and calculates the 6DoF pose of the controller and transfer it to the device driver through Serial/USB interface, then the device driver passes the data to the application to reflect the new calculated pose on the controlled object.

3.2.2. Device Driver Flowchart

A device driver is a particular form of software application that is designed to enable interaction with hardware devices. Without the required device driver, the corresponding hardware device fails to work. It acts like a translator between the lower-level hardware device and the higher-level application that uses the hardware.

Thus, in our project, we'll build a device driver using Microsoft Visual Studio program through C++ language. The function of this software is to drive the designed input device system by getting the transmitted data from the microcontroller of the system via Serial/USB connection, and then pass these data to a 3D application to apply 6DoF controlling. Figure 14 presents a flowchart that shows the general work methodology of our device driver.

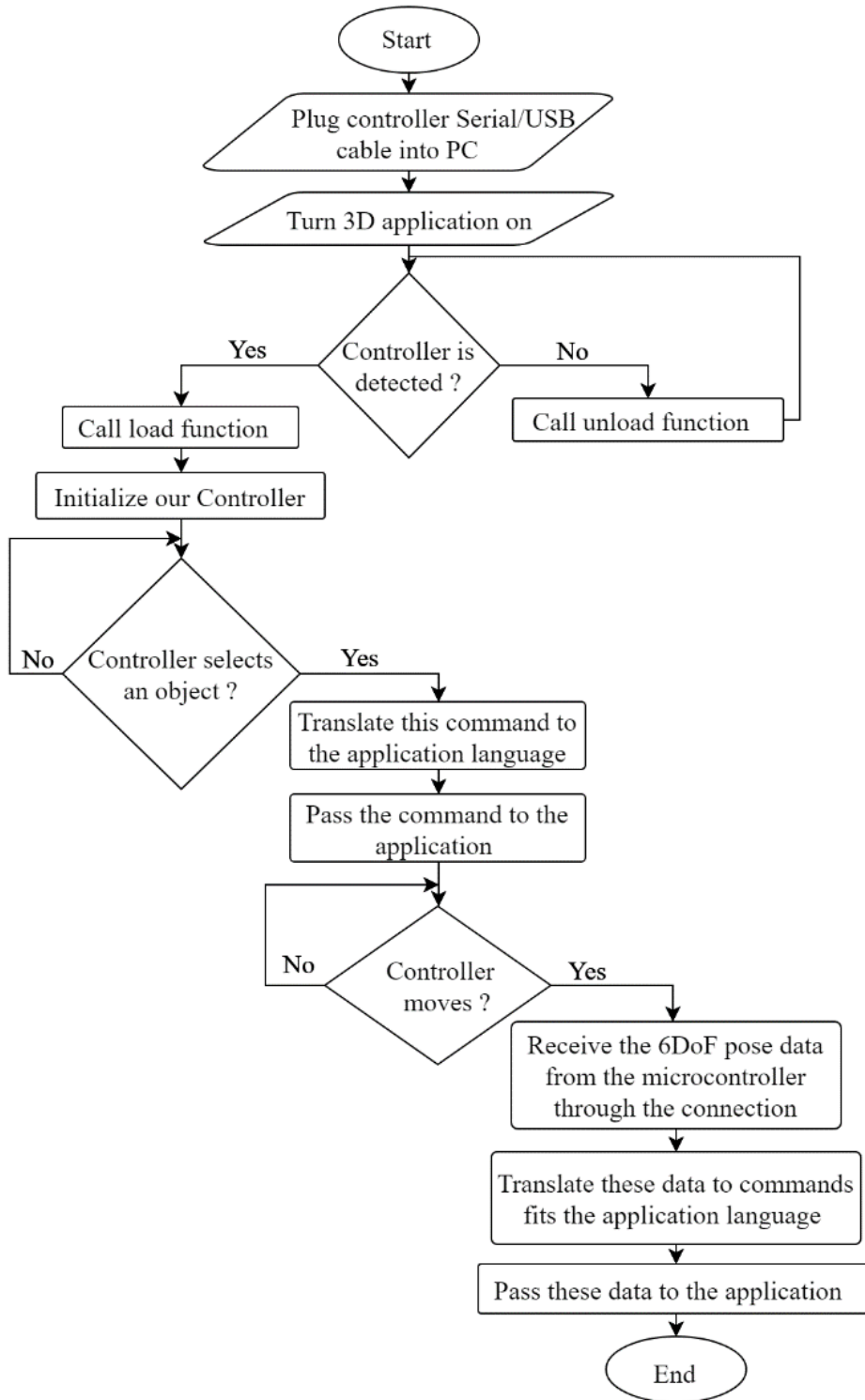


Figure 14: Device Driver Algorithm Flowchart

3.3. Diagrams

3.3.1. Block Diagram

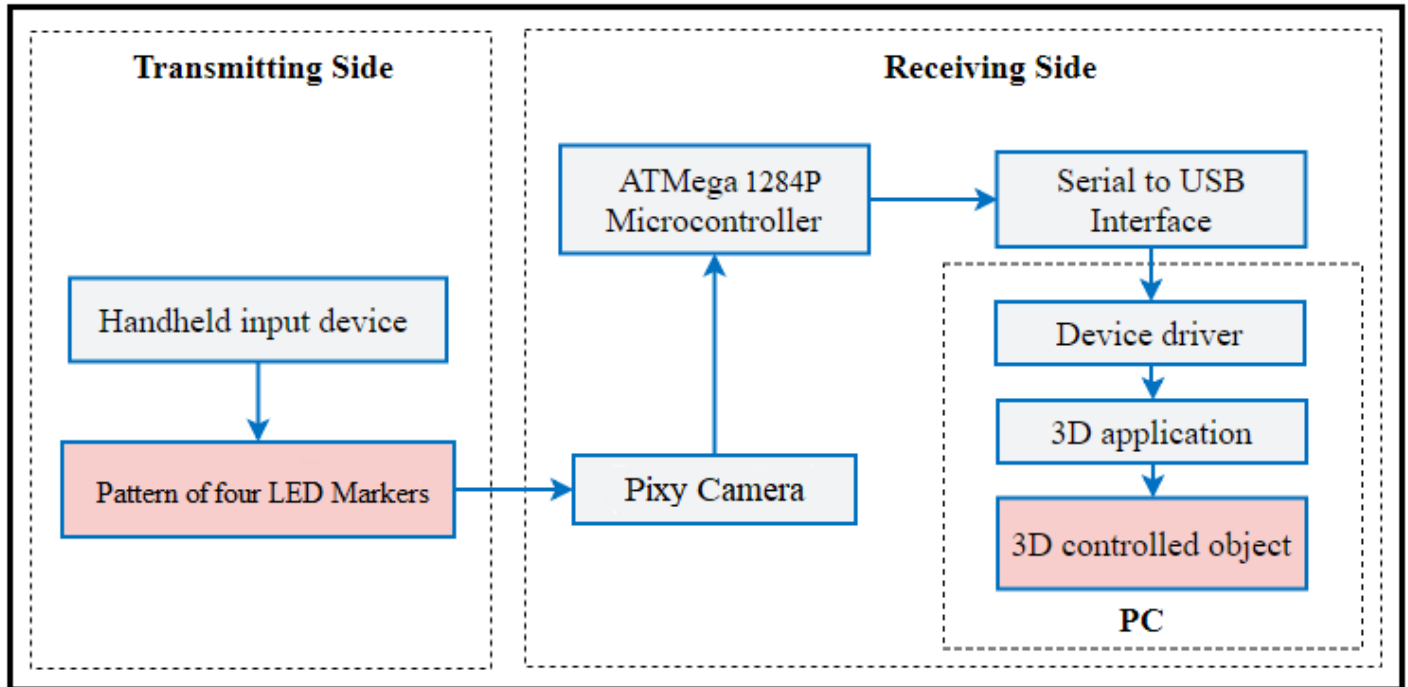


Figure 15: System Block Diagram

The block diagram in figure 15 shows how the system works. Handheld device box represents the input controller, Pattern of four LED markers box represents the Pattern mounted on the controller. So, if the user moves the controller in the working area of the camera, it means implicitly that he moves the pattern, so the Pixy camera tracks the movement of the markers, internally at camera built-in image processor, it calculates the pixel position for each tracked marker and then send these data to the microcontroller. After that, using these data the microcontroller calculates 6DoF pose of the controller by applying 3D pose estimation algorithms. Then, the new calculated 6DoF pose i.e. position (X, Y, Z) and orientation (Yaw, Pitch, Roll) will be transformed to the device driver by Serial/USB interface. On the PC, the device driver will pass the data to the 3D application to apply the new pose on the controlled 3D object.

3.3.2. Wiring Diagrams

3.3.2.1. Boot loading the ATmega1284P using Arduino Mega

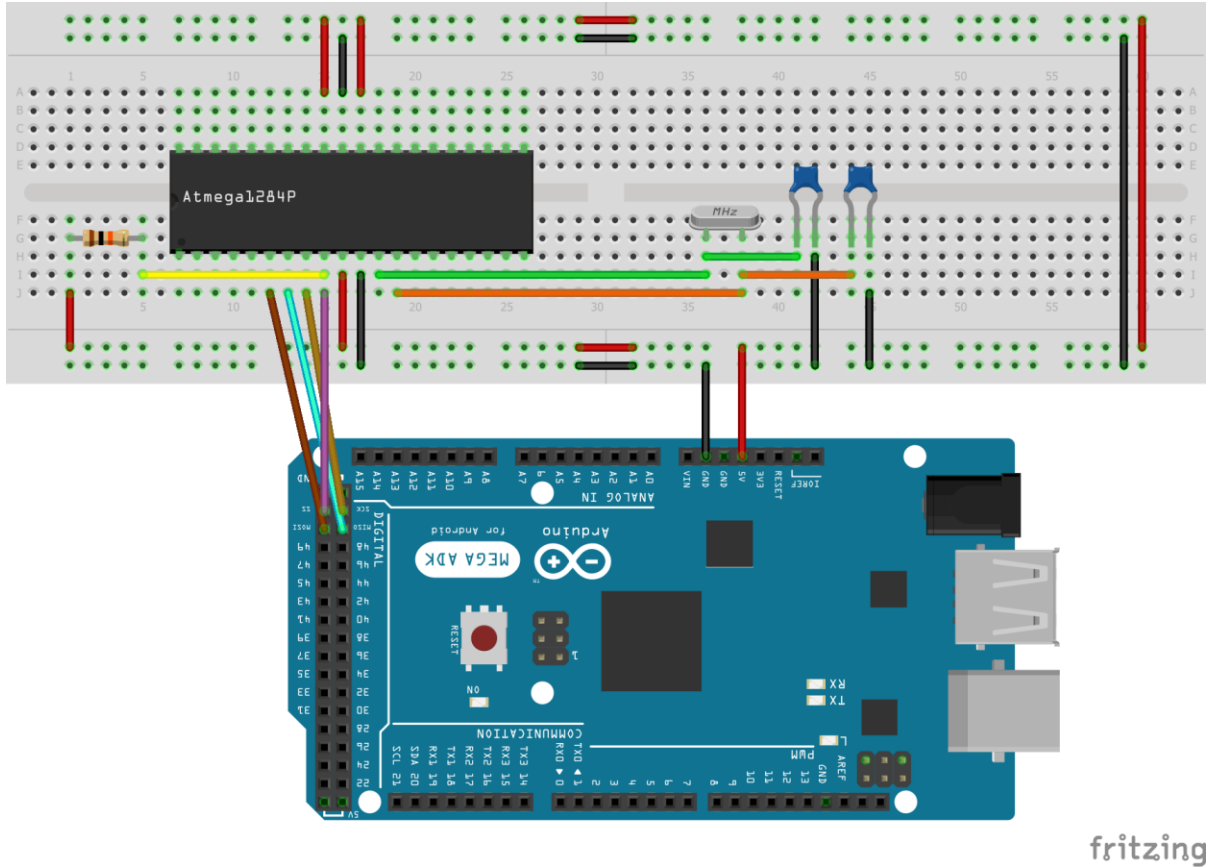


Figure 16: Connecting Atmega1284P with Arduino Mega for boot loading

Figure 16 shows how to connect Arduino Mega to ATmega1284P microcontroller for boot loading. Table 5 shows the pin map:

Table 5: Pixy Camera Pin Map with ATmega1284P

Arduino Mega Pin	ATmega1284P Pin
Pin50 – MISO	Pin5 -MISO
Pin51 – MOSI	Pin6 -MOSI
Pin52-SCK	Pin7 -SCK
Pin53– SS	Pin8 -SS

3.3.2.2. FTDI Serial/USB Interface to the ATmega1284P

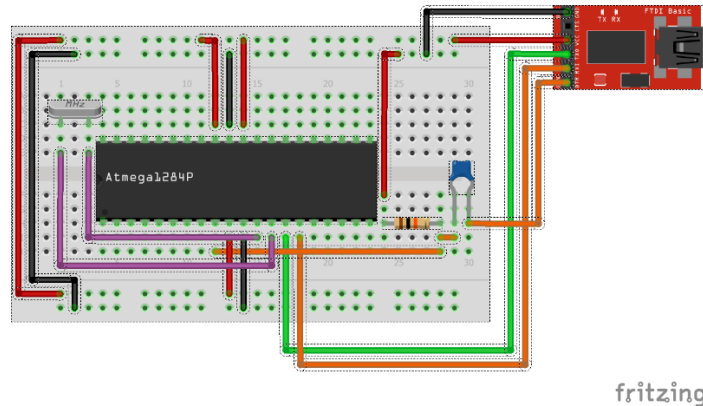


Figure 17: Interfacing FTDI Serial/USB to ATmega1284P

Figure 17 shows how to connect FTDI Serial/USB chip to the microcontroller. Table 6 shows the pin connection with some details.

Table 6: FTDI Serial/USB pin map with ATmega1284P

FTDI Serial/USB Pin	ATmega1284P Pin	Note
Pin1- GND	GND	-
Pin2 – CTS	-	It's not connected
Pin3 – VCC	VCC	-
Pin4 – TX	PD0 (RXD0)	At the microcontroller, when the USART0 receiver is enabled, this pin is configured as an input, it receives data from PC via FTDI TX pin.
Pin5 – RX	PD1 (TXD0)	At the microcontroller, when the USART0 transmitter is enabled, this pin is configured as an output, it sends data from the microcontroller to the PC via FTDI RX pin.
Pin6 – DTR	RESET	It's connected to the reset pin of the microcontroller via 0.1uF ceramic capacitor to make an auto reset circuit for the microcontroller.

3.3.2.3. Pixy Camera to ATmega1284P

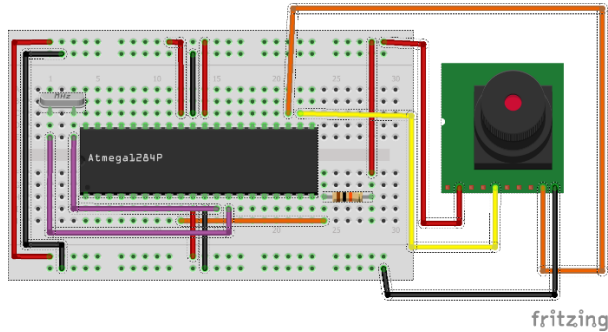


Figure 18: Pixy Camera to Atmega1284P

Figure 18 shows how to connect the Pixy camera to the ATmega1284P microcontroller.

Table 7 shows the pin map:

Table 7: Pixy Camera Pin Map with ATmega1284P

Pixy Camera Pin	ATmega1284P Pin	Note
Pin5 – SCL	PC0 (SCL)	SCL: Serial Clock line I2C Bus line
Pin9 – SDA	PC1 (SDA)	SDA: Serial Data line I2C Bus line
Pin2-VCC	VCC	-
Pin10– GND	GND	-

3.3.2.4. Whole System

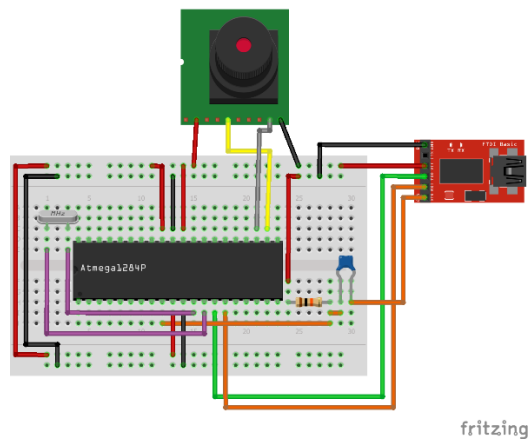


Figure 19: Whole System design

3.3.3. Schematic Diagrams

3.3.3.1. Bootloading ATmega1284P using Arduino Mega

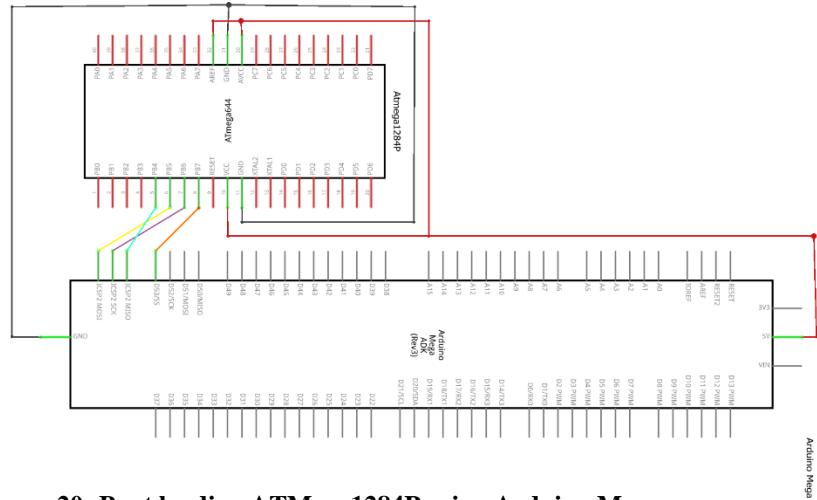


Figure 20: Boot loading ATmega1284P using Arduino Mega

fritzing

3.3.3.2. FTDI Serial/USB Interface to the ATmega1284P

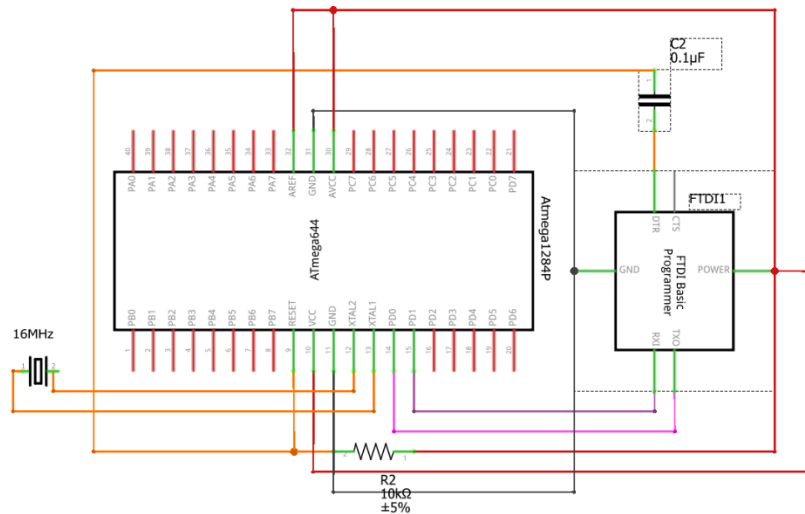


Figure 21: Schematic diagram of FTDI Serial/USB to ATmega1284P

fritzing

3.3.3.2.1. Pixy Camera to ATmega1284P

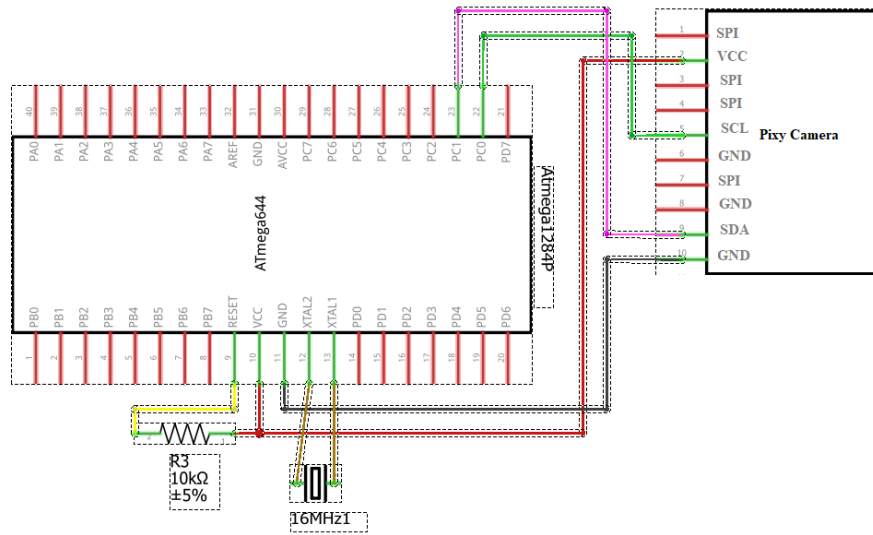


Figure 22: Schematic diagram of Pixy Camera to ATmega1284P

3.3.3.3. Whole System

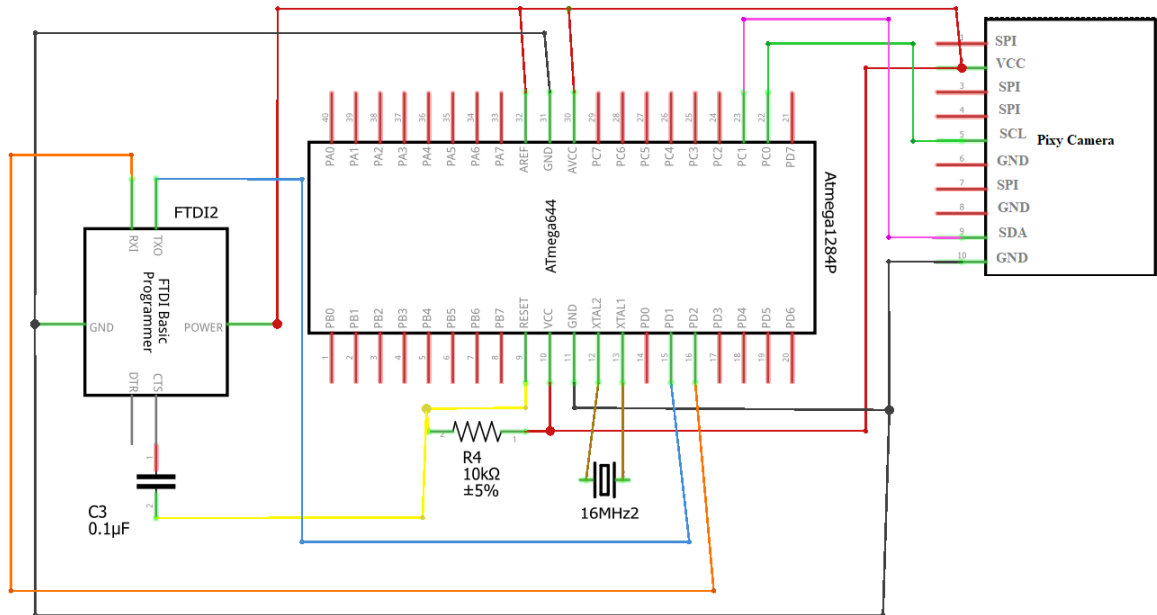


Figure 23: Schematic diagram of the whole system

Chapter 4

Hardware and Software Implementation

4.1 Overview

This Chapter describes the implementation of the hardware and the software for this project, such as the circuits connection and the software programs developed.

4.2 Hardware Implementation

As we previously explained that our project includes two separated hardware sides, The transmitter and receiver sides, so we implemented each of these sides as follows:

4.2.1 Receiving side Implementation:

- **Step1:** as shown in figure 24 We burn a bootloader on Atmega1284P MCU since it doesn't have one, this step is basically the start because each MCU must be successfully boot-loaded to get ready for receiving data and programs and make processing as expected. The steps we followed to burn the bootloader on Atmega1284P using Arduino Mega as ISP are as follows:

- a. Install Arduino 1.0.6 IDE.
- b. Install mighty1284P platform for the Arduino IDE and adding it to this path "C:\Program Files (x86)\Arduino\hardware\mighty-1284p".
- c. Upload the "ArduinoISP" Sketch into Arduino Mega.
- d. Select Tools>Boards> Original Mighty1284P 16MHz External.
- e. Choose Tools>Programmer>Arduino as ISP
- f. Select Tools>Burn Bootloader.

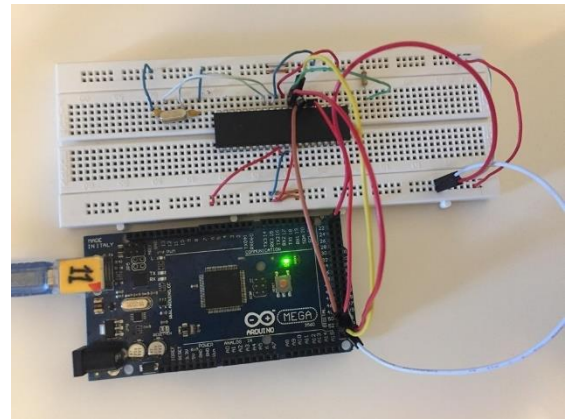


Figure 24: Implementation of Atmega1284P Bootloading

- **Step2:** We connect the FTDI chip to the Atmega1284P MCU via Rx and Tx pins to make a connection between the MCU and the device driver software.
- **Step3:** We connect the Pixy Camera to the Atmega1284P MCU via I²C bus [using SDA and SCL pins].
- **Step4:** Finally, we integrate all the components by soldering them on 7cm*9cm PCB piece as shown in figure 25.

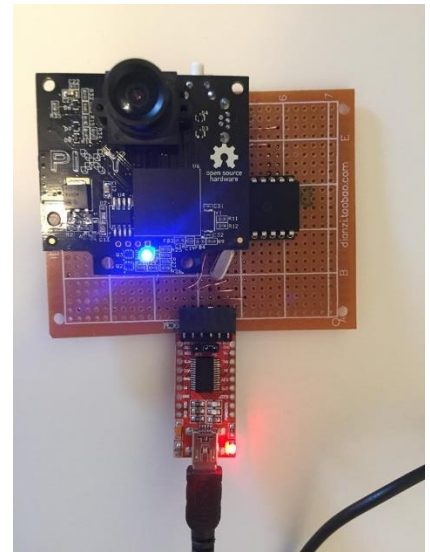


Figure 25: All components together

Result: After these steps, we got a whole receiving side that is ready to detect the markers via the pixy camera, make the calculations at the MCU and send data via FTDI chip.

4.2.2 Transmitting side Implementation:

- **Step1:** We design the pattern of the four LED Markers considering these points:
 - a. **The front distance of the pointer that is holding the markers,** approximately, we have [4 - 4.5]cm that can hold the markers, this distance is enough and practical.
 - b. **The size of the LEDs.**
 - c. **The brightness of the LEDs,** because the LEDs are chosen to have high brightness to be easy detectable by the camera from more than 1 meter, because of that, the LEDs must be far enough from each other to avoid problem shown in figure 26.
 - d. **The best arrangement of the LEDs by their colors,** i.e., [Red, Blue, Green, White] arrangement is better than [Red, White, Blue, Green] because red and white lights are bit alike so the pixy camera sees them as one signature.
 - e. **The geometrical arrangement of the markers criteria.**

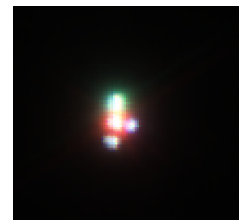


Figure 26: Brightness Problem

Result: Taking the previous points into consideration, we come up with the design of the pattern.

- **Step2:** We connect the 4 LEDs to a dip switch and then to the lithium Ion battery to avoid the Inefficient continuous lighting of the LEDs.
- **Step3:** Finally, we integrate all the components by soldering them on 5cm*7cm PCB piece and hock it into the pointer as shown on figure 27.



Figure 27:Front side of the pointer

Result: After these steps, we get a transmitting side that is a pointer with pattern of LED markers that can be detectable by the pixy camera.

4.3 Software Implementation

In our project, we implemented three programs, the first one that's run on the Atmega1284P MCU, the second one is a device driver for interfacing the hardware to the computer and the last one is a 3D graphical application for testing. These programs are developed as follows:

- **Core Program:** this program is uploaded on the MCU to run as the core of the project, it generally does the following tasks:
 - 1- Markers detection:

As we previously mentioned, the pixy camera detects the signatures of the tracked LEDs [i.e. each LED marker assigned a signature from 1 to 4], then it returns the X, Y coordinate of the center of the detected signature, width and height of the detected signature.
 - 2- Pattern detection:

In this section of the code, we have to filter the results of the markers detection step since the camera can easily detect any unrelated LED, this is called a false positive detection [i.e. Return many occurrences for the same signature due to noise in the surrounding environment lights.] to overcome this problem, we suggest a criteria to decide if the camera sees the desired pattern or not, the

received signatures must pass the criteria conditions to consider as a desired pattern. The criteria conditions are:

- a- If there are just four different signatures at one time with the desired arrangement of the signatures and the appropriate distance between them then those are formed the desired pattern.
 - b- If there are just four different signatures at one time with the desired arrangement of the signatures but without appropriate distance between them then those are not formed the desired pattern, ignore them.
 - c- If there are just four different signatures at one time with appropriate distance between them but with different arrangement then those are not formed the desired pattern, ignore them.
 - d- If there are more than four signatures at one time, loop on each four of them until you find the desired pattern, this loop has many if statements to reduce its complexity.
- 3- 6DoF Pose detection: This section is considering the core, here we calculate the 6DoF of the pointer using mathematical calculations on the detected pattern markers values.

Generally, this problem is called PnP problem (Perspective-n-Point) is the problem of estimating the pose of an object given a set of n 3D points in the world and their corresponding 2D projections in the image. The object pose consists of 6 degrees-of freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. In our project $n=4$, we used a combination of Euler equation and angels to calculate the pose.

- 4- Communication with the driver.

- **Windows Client Device Driver:** this program is developed from scratch for interfacing the hardware circuit to the operating system, it's very important to mention that this program does nothing of the mathematical calculations of pose detection, it just receives the data from the MCU via the FTDI and then communicate with the 3D graphical application.

In details, it does the following:

- 1- Detects the registered hardware when it's plugged into the PC, mainly this happens by two basic functions,
 - a. **IoCreateDevice** to create a device object and attaching it to the device stack.
 - b. **IoRegisterDeviceInterface** to register a device interface class and to create an instance of the interface for that device.
- 2- Receives the data from the device, this done by the **DeviceIoControl** method that build a communication protocol called IOTCL (Input and Output Control) that control the access rights and the buffers use.
- 3- Communicates with the 3D graphical application, as explained on the previous step, this step also done with the same method.

Note: Device Driver development is a big topic, in our development we followed the basics of building the driver, but basically for developing and debugging our driver we use a virtual machine running windows 7 because debugging the driver means debugging the operating system itself and that makes the OS freezes which is unpractical for the computer that hosting the source code of the driver .

- **3D Graphical application:** this desktop program is developed from scratch using Qt and OpenGL libraries integrated in Visual Studio 2015, it receives the data from the device driver software and reflect it on the controlled 3D objects, figure 28 shows the GUI of it.

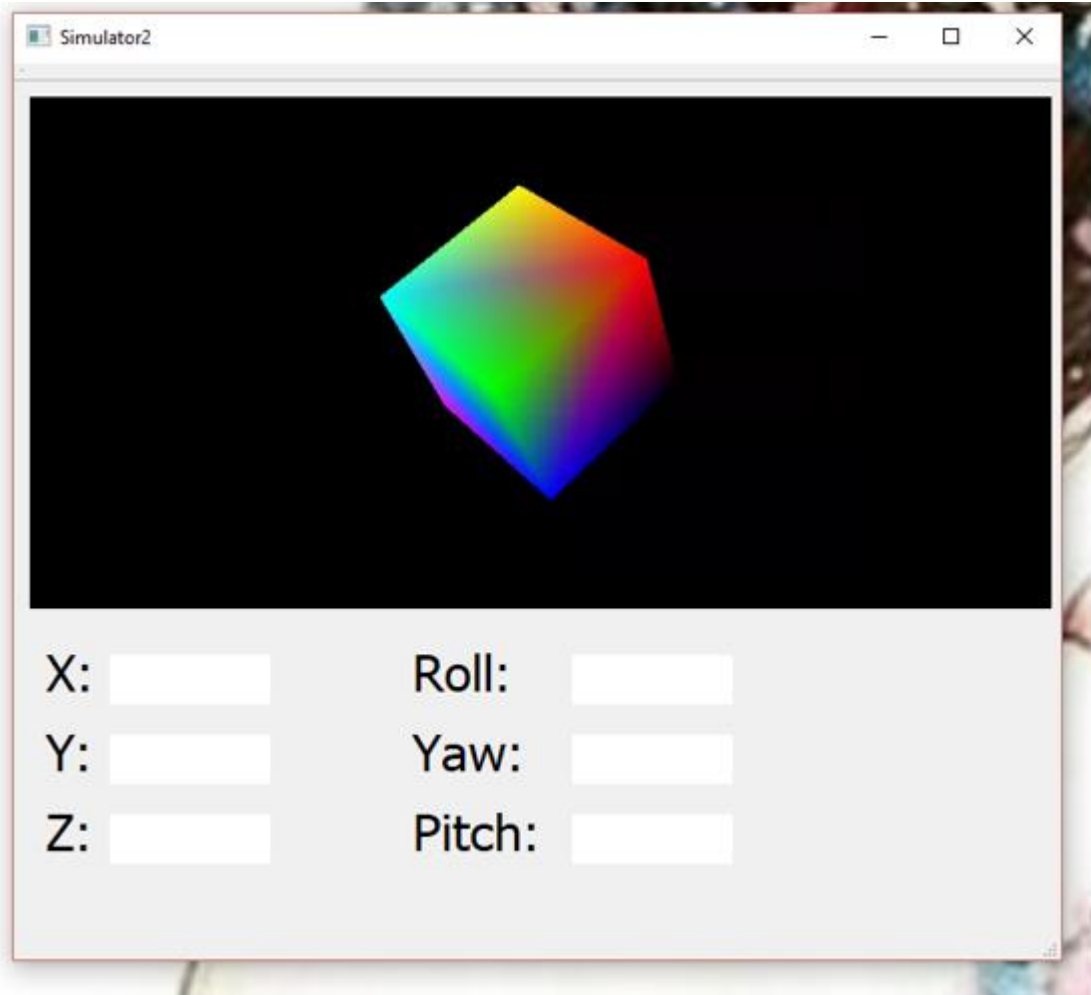


Figure 28: GUI of the 3D Graphical application

4.4 Implementation Issues

- **Issue 1:** trying to connect the FTDI chip to the Atmega1284p for transmitting and receiving data continuously gives “FTDI FT232R not working - stk500_getsync(): not in sync: resp=0x00” Error.

Solution: remove the two 22pF capacitors that were connected to the external 16MHz crystal to the Atmega1284P MCU solves the problem.

- **Issue 2:** The Pixy Camera doesn't see the LED markers obviously at distance more than, approximately, 1 meter.

- Solution:** change the LEDs to another LEDs with higher brightness, this makes the camera sees the LEDs at distance nearly up to 2.5 meters.
- **Issue 3:** Increase the brightness of the LEDs, rises the problem shown in figure 26.
Solution: we decided to change the LEDs to smaller and brighter LEDs.
 - **Issue4:** Compilation errors arise through device driver development using visual studio 2017.

Solution: several things solve these errors:

- 1- Uninstall Visual Studio 2017 and delete all related files.
- 2- Install visual studio 2015.
- 3- Install Windows Software Development Kit (SDK) – Windows 10.0.10586.0.
- 4- Install Windows Driver Kit (WDK) – Windows 10.0.10586.0.

Note: SDK and WDK must be the same version otherwise compilation error will arise again.

4.5 Implementation Results

By the end of the implementation process, the 6DoF input device shown in figure 29 was constructed successfully.



Figure 29: 6DoF Input device

Chapter 5

Validation and Testing

5.1 Overview

In this Chapter, we will discuss the testing of all components of the system and the results obtained, we test all the parts to ensure that all of the functions work perfectly and without errors.

5.2 Hardware Testing

5.2.1 Testing Atmega1284P MCU:

After uploading the bootloader to the Atmega1284P, we must test if it works fine, so the test was to upload simple blinking program to the Atmega1284P MCU using the Arduino Mega as follows:

- a- Connect Arduino Mega to the PC and uploading the Arduino as ISP sketch.
- b- Connect the Atmega1284p to the Arduino mega as shown in figure [30].
- c- Connect LED to Atmega1284p Digital pin.
- d- Upload simple blinking code to the Atmega1284P using (Upload using programmer) option in Arduino IDE.

Result: The LED blinked as expected.

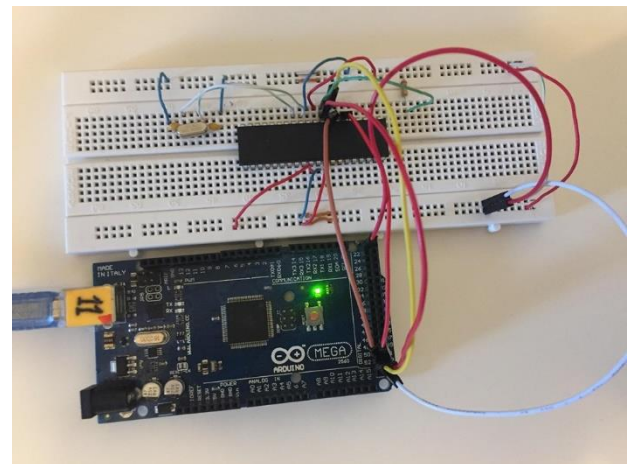


Figure 30: Testing the bootloader

5.2.2 Testing the FTDI chip with the Atmega1284P MCU:

We connect the FTDI to the Atmega1284P as shown in figure[31] and trying to upload simple blinking code with another delay value.

Result: The LED blinked as expected.

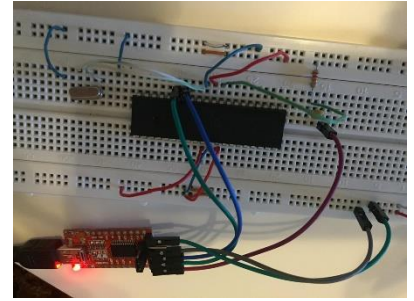


Figure 31: Test FTDI

5.2.3 Testing the Pixy Camera:

We Installing the PixyMon Software to test the working of the camera, this software allows the user to monitor the camera image, and we put different colors of the LEDs and assign signatures to each one.

Result: The camera detected the LEDs and output the X, Y coordinates of the center of each signature as expected

5.3 Software Testing

5.3.1 Testing the core code:

- a- Marker Detection Testing: in this test we tried several experiments of moving the markers to the rightmost and to the leftmost, also we tried to move the markers far away from the camera, table [8] shows the approximate results:

Table 8: Marker Detection Testing Results

Marker Color	Possibility of false positive	Could be detected by the camera from
Red	High	2.5 meters
Blue	Very Low	1.5 meter
Green	Low	1.5 meter
White	Very high	2.5 meters

b- **Pattern detection Testing:** in this test we tried to keep the desired LEDs pattern (pointer) plus moving another LEDs with different distances around it.

Result: The code works pretty fine and ignore any surrounding unrelated LEDs.

c- **6DoF Pose Detection Testing:** in this test we made several experiments trying to move the pointer in front the camera to get it's pose, we considered each user movement of the pointer as a one experiment, by taking the results of set of 10 experiments, we got these results:

1- Considering the speed of response to user movements: If the user moves the pointer very quickly, i.e. without any delay between movements, number of the right pose detection of the pointer become less.

2- Considering the distance of the movements: The system works perfect in range of 1 meter from the camera, when the distance become larger the system accuracy becomes less.

3- Considering the reflection of the movement for each coordinate :

Degree of freedom	#of right reflection out of 10	Note
X	10	The system always reflects the object's X coordinate correctly.
Y	10	The system always reflects the object's Y coordinate correctly.
Z	9	The system reflects the object's Z coordinate correctly.
Roll	6	The system reflects the object's Roll angle correctly in most trials.
Yaw	2	The system doesn't reflect the object's Yaw angle correctly in most trials.
Pitch	9	The system reflects the object's Pitch angle correctly.

Table 9: 6DoF Pose Detection Testing

- d- **Sending Data to the driver testing:** This code segment works fine, the microcontroller was able to be synchronized in sending the (X, Y, Z) and (Roll, Yaw, Pitch) data to the device driver every 10 frames.

5.3.2 Testing the device driver:

In this test, we tried to plug and unplug the hardware circuit from the PC, as a result, every time we unplugged the hardware circuit the device driver shows that the device is unregistered and stopped sending data to the 3D graphical application.

5.3.3 Testing the 3D graphical application:

In this test, we tried to sending data from the driver to the application, as a result, the application works fine in receiving data, but not reflecting it on the controlled object accurately.

5.4 System Testing

To test the overall system “final construction of receiving and transmitting sides and the applications “, this includes checking all the following points:

- 1- Check the validity of the device driver for communication with the device and the 3D graphical application.
- 2- Check the validity of the 3D graphical application for reflecting the movement of the pointer on the 3D controlled object.
- 3- Check the transmitting LEDs on the pointer.
- 4- Check the receiving Camera and the calculations on the MCU.

This come up with reflecting the user movement of the pointer on the controlled 3D object on the 3D graphical application.

Chapter 6

Conclusion

6.1 Summary

In this project, we constructed a 6DoF input device that allows the expert or inexperienced users to control 3D objects in up to 6DoF.

The 6DoF input device that includes two sides, transmitting side that has a pattern of LED markers and receiving side that has the camera and the microcontroller interfaced to the PC using device driver, so when the user moves the pointer that has the pattern of markers, his movement will be reflected on the controller 3D object into the developed 3D application with 6DoF.

At the end of the project, we believe that this project would be an added value to the graphics market and take a role of creating a new human interaction with the 3D environment with this affordable and easy to use input device.

6.2 Challenges faced

We faced some issues, but the hardest issue that we've faced is with the Wii IR Camera, it's not available in our local market, it doesn't have a manufacture datasheet also it doesn't have a wide resource on the internet, so it was very hard to deal with.

We faced another issue with the new alternative camera, pixy camera, it's not working very well with lights, so it returns a lot of false positive cases that makes the calculations harder.

6.3 Future works

Ultimately, with this project, we aim to suggest a new affordable 6DoF input device for expert and inexperienced users. To achieve this, several features could be improved. Mainly the system would be more efficient if it is built with Wii IR Camera, also, the system could connect to the PC using Wi-fi connection.

References

- [1] Space Mouse Pro. April 20, 2018, Retrieved October 20, 2018 from <https://www.3dconnexion.com/products/spacemouse/spacemousepro.html>.
- [2] Space Mouse Wireless. July 5, 2018, Retrieved October 20, 2018 from https://www.3dconnexion.com/spacemouse_wireless/en/.
- [3] Spaceball 5000. August 23, 2003, Retrieved October 20, 2018 from http://spacemice.org/pdf/SpaceBall_5000.pdf.
- [4] The Daydream 6DoF controllers. December 1, 2018, Retrieved November 2, 2018 from <https://developers.google.com/vr/experimental/6dof-controllers>.
- [5] Optical Tracking. November 2, 2018. Retrieved December 10, 2018 from https://en.wikipedia.org/wiki/Positional_tracking#Optical_tracking.
- [6] ATmega644P/V Datasheet. October, 2016. Retrieved December 1, 2018 from https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42744-ATmega644P_Datasheet.pdf.
- [7] Wii IR Camera. December 15, 2014. Retrieved December 1, 2018 from http://wiibrew.org/wiki/Wiimote#IR_Camera.
- [8] Pixy Camera. May 14, 2012. Retrieved May 12, 2018 from <http://www.cmucam.org/projects/cmucam5>
- [9] FT232R Chip. August 7, 2010. Retrieved December 5, 2018 from <https://www.ftdichip.com/Products/ICs/FT232R.htm>.
- [10] VisualDDK Tool. June 19, 2009. Retrieved November 24, 2018 from <http://visualddk.sysprogs.org/>.