

# **TABLE OF CONTENTS**

1. Abstract
2. Introduction
3. Related Works
4. H/W & S/W Requirements
5. Flowcharts
6. Proposed Work and Implementation
7. Output Screen Shots
8. Results and Discussion
9. Bibliography

## 1. Abstract

Microcontrollers are a very basic need in today's technology. In fields like Internet of Things (IOT) and embedded systems microcontrollers act like a processing unit in some sense. Microcontrollers are responsible for the actual working of the component, error checking, and as probable intermediary between many components. Thus, the microcontroller is a self-contained system.

Current microcontrollers like Arduino board, BeagleBone board and raspberry pi to name a few have their own processing units, memory and peripheral component connections to it. While a microcontroller does not have enough power as a microprocessor, it makes up for that lack of power in more functionality as a single unit. Also many microcontrollers are very cheap and affordable and many students and hobbyists use them to create DIY projects and learn about the internals of the system.

Because microcontrollers are so widely used it only makes sense that the software that controls the microcontroller's behaviour is always updated. Obsolete drivers can create a load of issues such as being unable to interface the microcontroller properly, being unable to sense the microcontroller, etc. to name a few. Thus, the device driver and software of the microcontroller must always be updated and managed. Device driver act as an interface to the microcontroller itself and is used to program it. Any IDE that wants to upload or flash the created program (code) to a microcontroller must do so through the device driver.

## 2. Introduction

The update delivery system allows the management of the above driver issue by checking for updates. This system however checks for update only through the user's permission and will notify the user accordingly.

The actual version and the software update itself is hosted on the server and the current version either is put as a configuration file or is extracted directly through the microcontroller when it is connected. The versions are compared and if the server's version has a greater version then it means that a new version has been uploaded to server and must be downloaded. This is how the software updates are managed.

The program has three parts:

**Part 1:** Checks whether the device is connected or not

This is important, as the user may want to update the device as soon as it has been downloaded. While checking, the version is also extracted and the COM port at which the device is connected is outputted.

**Part 2:** Checks and compares the current version and the hosted version

This is effectively the part where it is decided whether the update is necessary or not. If the versions are equal in number, it means that the driver is up-to-date. If the server's version (hosted version) is greater, then a new version is available.

### **Part 3: Downloads the software-update**

This is perhaps the easiest implementation. This step simply downloads the file that has the software update. Since all the comparisons have been done beforehand this step does not involve any computation.

The above three parts summarise our project. This is an implementation done using C programming language and uses Windows API. Thus, this is not fully cross-platform. Since the Microsoft Windows operating system is used by 98.91% of the population today, we have decided to target this operating system's implementation.

## **3. Related Works**

**Null Com Emulator for Windows**: Allows echoing (Outputting) information to one port and reading that information to another port. Allows us to emulate the presence of a microcontroller when there is none (Windows only).

**Devlib**: An API developed in python that allows communication between host and remote devices such as mobile phones, tablets that run a linux-based operating system. Supported on both Windows and Linux.

**Ktechlab**: An IDE for development of software for microcontrollers and electronics. Supported for linux only.

**Arduino IDE**: An IDE for development of software for the Arduino microcontroller. Supported on both Windows and Linux.

**Arduino-Thread**: A C library for using threads in the Arduino microcontroller. Used as a C header file.

## **4. Hardware and Software Requirements**

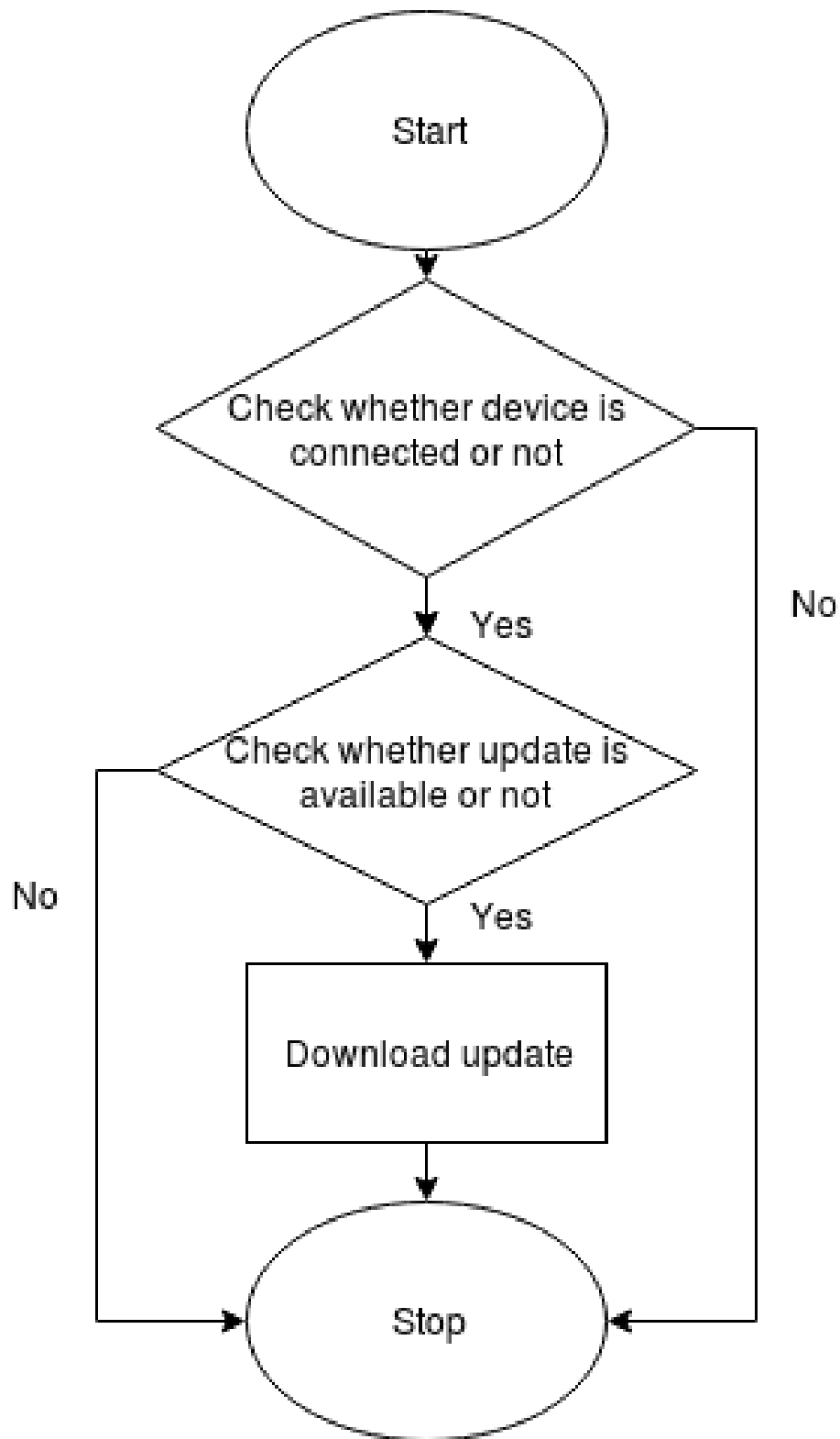
### **Hardware:**

1 GB of RAM  
1 GB of Hard disk's available space  
Network Connectivity (To communicate with the server)  
Arduino UNO Board

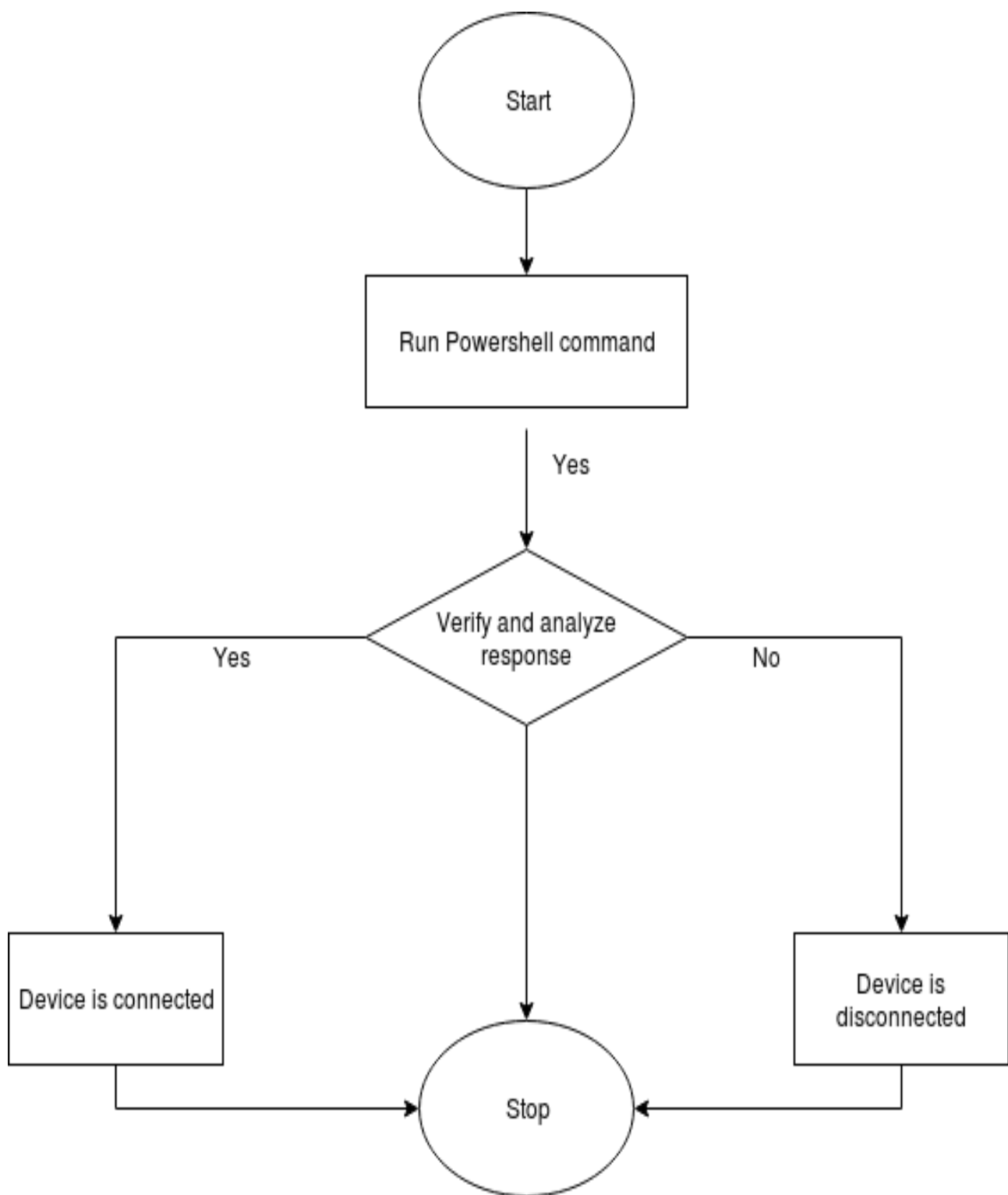
### **Software:**

Windows Operating System  
Windows PowerShell  
Null Com Emulator for Windows

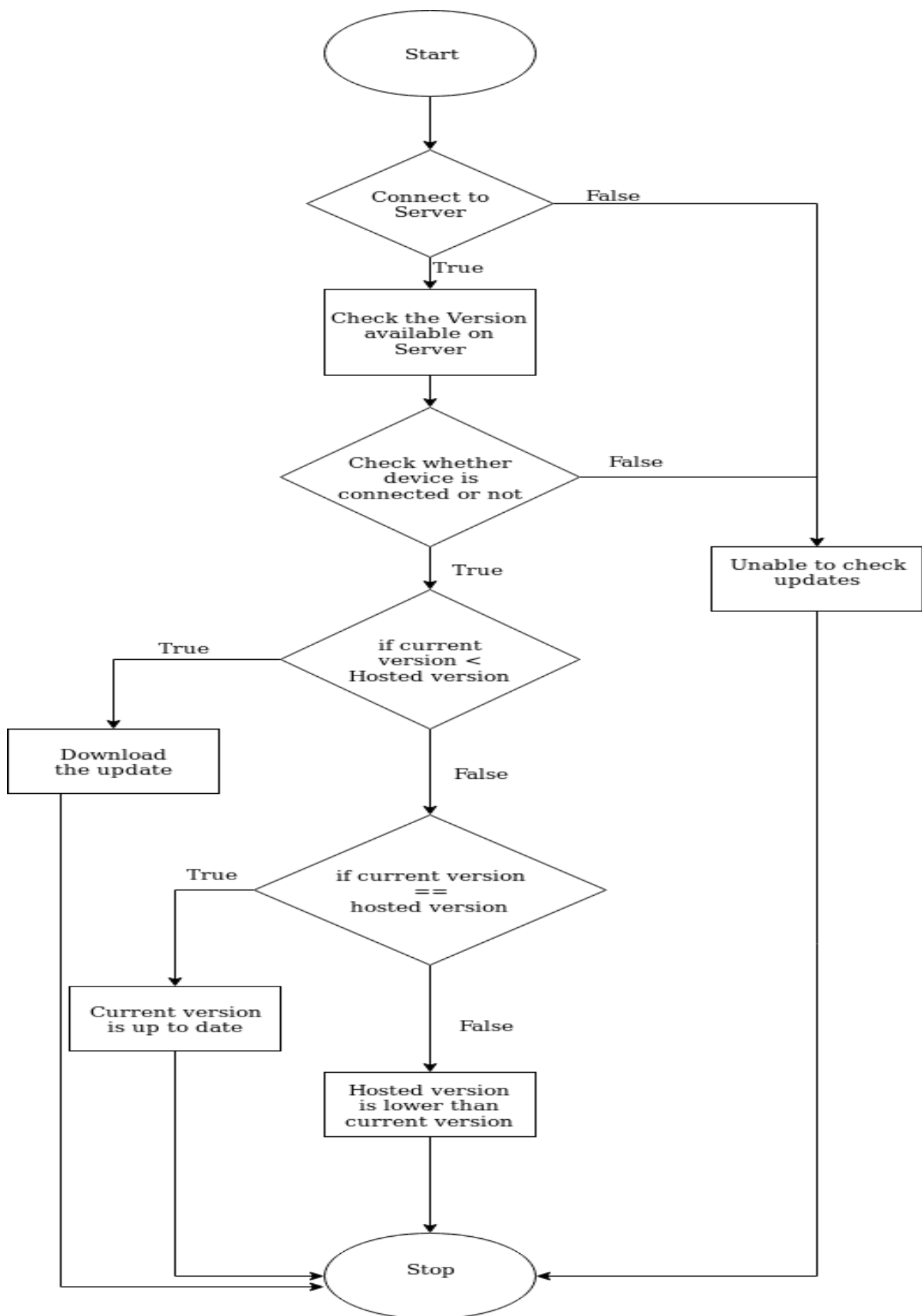
## 5. Flowcharts



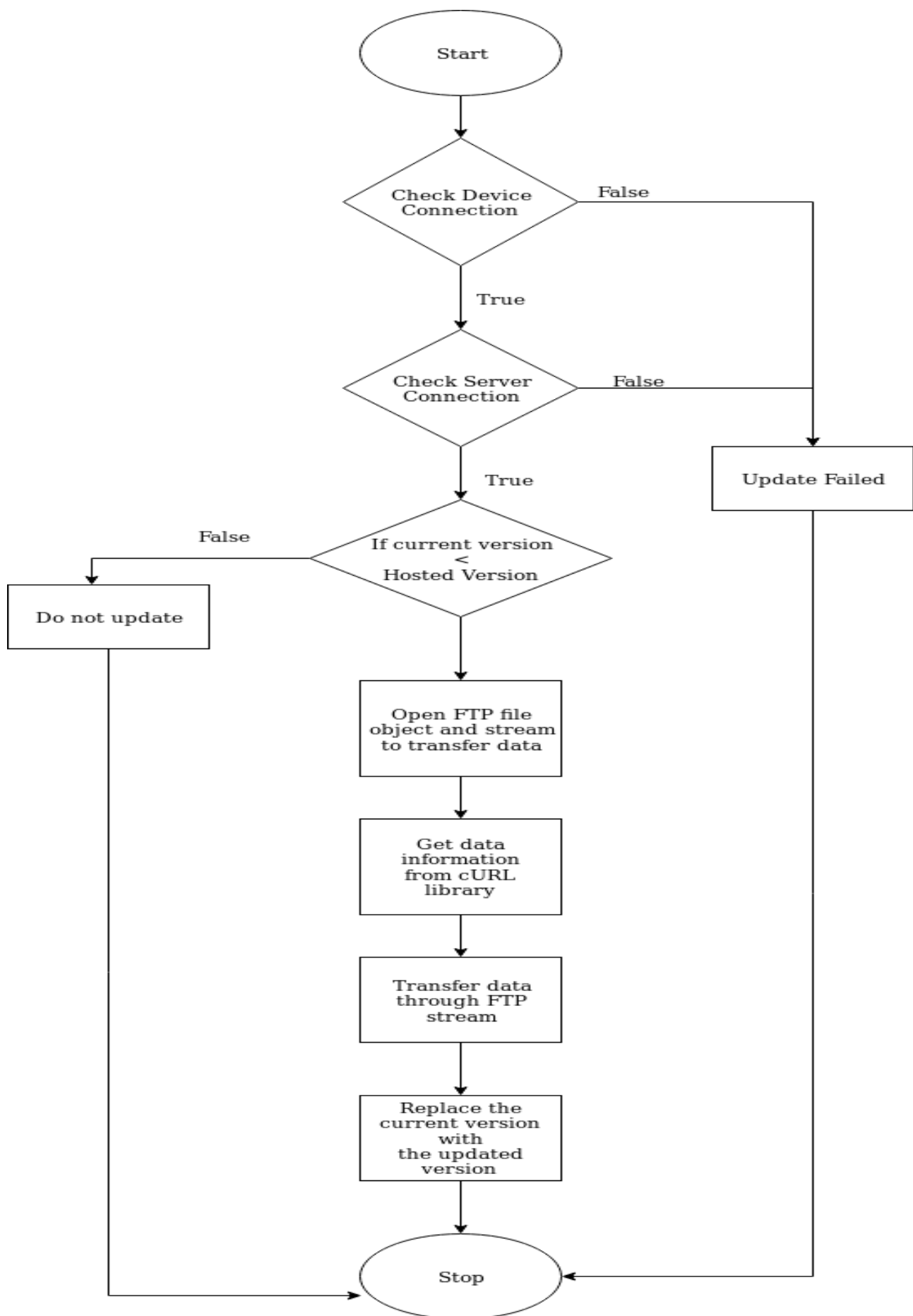
Main Flowchart



Check status of device



Check and compare versions



Download the updates

## 6. Proposed Work

In our project we have used the curl library. The library is represented by the header file '<curl/curl.h>' in the code. This library allows fine manipulation of the requests sent through the network so it is of paramount importance if we want to send custom requests. Also because our server is custom made and very simple we can get the output directly in the command line using the curl tool. The curl C library is simply an interface to this tool.

We have also defined two custom header files named "<custom.h>" and "<str.h>". "<custom.h>" has a custom defined data type called status which has two members named port and stat. "<str.h>" similarly has a custom defined string data type having pointer to the string and the length of that string. We have defined string data type as the C programming language does not have a string data type by default and the server responses are easier to parse as strings.

For this Proof of Concept we have chosen the Arduino UNO microcontroller and thus have defined it in the "<custom.h>" header file.

We will now explain each part of the implementation with code snippets.

### **Part 1:** *Check whether the device is connected or not*

Since there is no direct or tested API to communicate with the Arduino we have used a PowerShell command to open a connection to the device. The communication stream is opened only if the device is connected. PowerShell in Windows is much more powerful version of the command prompt that also has scripting functionality. The snippet of code is as follows:

```
-----CODE-----
#include "custom.h"

string connection_status(char *devDesc)
{
    char *cmd1 = "cmd /Q /C FOR /F \"tokens=* USEBACKQ\" %F
                IN (`powershell \"Get-WMIObject
                Win32_SerialPort|Where-
                Object{$_<Description -eq '\";char *cmd2 = '\"}
                | Select-Object -expand DeviceID\"`) DO
                (echo %F)";

    string command = string_concat((char *[])
    {
        cmd1,devDesc,cmd2
    },3);

    string response = process(command.ptr,"rt",FALSE);
    response.ptr[strlen(response.ptr)-1] = EOS;
```



```

if(strcmp(response.ptr,"") == 0)
{
    printf("\nDevice is disconnected\n");
    return (string)
    {
        NULL,0
    };
}
else
{
    printf("\nDevice is connected at %s\n",response.ptr);
    return response;
}
}

```

-----END-----

The above snippet demonstrates the need for the string data type. cmd1 and cmd2 are the commands that are used to check the microcontroller connectivity. The process() call allows us to spawn a process with a particular command. The microcontroller responds with a stream of data, which is accepted as string. In case the received characters compare to NULL then we conclude that the device is disconnected. However if some data was received then the device is connected and is responding.

## **Part 2: Check and compare the version on the Arduino and the version hosted on server**

The following code is responsible for fetching only the version on the server and getting the version running currently on Arduino and comparing them. Since version numbers are floating point numbers i.e. 1.0, 10.0, 2.1, etc. we need to convert the received data to float from string. If this step is successful then the update can be downloaded directly. All the computation needed to decide whether download is necessary or not is done in the following code.

-----CODE-----

```

#include <curl/curl.h>
#include "custom.h"

size_t writefunc(void *ptr, size_t size, size_t nmemb, string *s)
{
    size_t new_len = s->len + size*nmemb;
    s->ptr = realloc(s->ptr, new_len+1);

    if (s->ptr == NULL)
    {
        fprintf(stderr, "realloc() failed\n");
        exit(EXIT_FAILURE);
    }
}

```

```

    }

    memcpy(s->ptr+s->len, ptr, size*nmemb);
    s->ptr[new_len] = EOS;
    s->len = new_len;

    return size*nmemb;
}

string get_version(char *addr)
{
    string response;
    init_string(&response,0);

    CURL *curl;
    CURLcode res;

    curl = curl_easy_init();

    if(curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, addr);
        curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, writefunc);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);

        /* Perform the request, res will get the return code */
        res = curl_easy_perform(curl);

        /* Check for errors */
        if(res != CURLE_OK)
        {
            fprintf(stderr, "\ncurl_easy_perform()
failed: %s\n", curl_easy_strerror(res));
            response.ptr = NULL;
            response.len = 0;
        }

        /* always cleanup */
        curl_easy_cleanup(curl);
    }
    return response;
}

status check_updates(char *addr, char *desc)
{
    printf("\nChecking for updates !!\n");

```

```

string res = get_version(addr);
status s = {NULL,0};
if(res.ptr != NULL)
{
    printf("\nHosted Version: %s",res.ptr);
    double server_ver = atof(res.ptr);

    printf("\nObtaining Current Version from Device (make sure
the device is connected) !!!\n");
    res = connection_status(desc);

    if(res.ptr == NULL)
        return s;

    string ch = readSerial(res.ptr);

    if(ch.ptr != NULL)
    {
        printf("\nCurrent Version: %s\n",ch.ptr);
        double client_ver = atof(ch.ptr);

        if (server_ver == client_ver)
        {
            printf("\n[*]Current Version is up to date\n");
            return (status)
            {
                res.ptr,-1
            };
        }
        else if (server_ver > client_ver)
        {
            printf("\n[-]Current Version obsolete. New Version
Available\n");
            return (status)
            {
                res.ptr,1
            };
        }
        else
        {
            printf("\n[+]Hosted Version is lower than current
version\n");
            return (status)
            {
                res.ptr,-1
            };
        }
    }
}

```

```

        }
    }
    else
    {
        printf("\nFailed to read the current version from the
device\n");
        return s;
    }
}
else
{
    printf("\nFailed to get the hosted version from the
server\n");
    return s;
}
}

```

-----END-----

The check\_updates() function above returns the value in terms of structure status defined in the “<custom.h>” header file. Since this structure variables do not have any kind of access specifiers the members can be accessed freely from any program which has this header file included. The next step depends on this code to download the update.

### **Part 3: Download the software-update**

This step simply downloads the software update. All the decisions have been made and computed in part 2 as defined above. The update should only be downloaded if the current version is outdated. Here we are using a combination of the FTP (File Transfer Protocol) and cURL library to download the file.

-----CODE-----

```

#include "custom.h"
#include <curl/curl.h>

/* <DESC>
 * Get a single file from an FTP server.
 * </DESC>
 */

struct FtpFile
{
    const char *filename;
    FILE *stream;
};

```

```

static size_t my_fwrite(void *buffer, size_t size, size_t nmemb, void
*stream)
{
    struct FtpFile *out = (struct FtpFile *)stream;
    if(out && !out->stream)
    {
        /* open file for writing */
        out->stream = fopen(out->filename, "wb");
        if(!out->stream)
            return -1; /* failure, can't open file to write */
    }
    return fwrite(buffer, size, nmemb, out->stream);
}

```

```

int download(char *addr)
{
    CURL *curl;
    CURLcode res;
    struct FtpFile ftpfile =
    {
        "package.hex", /* name to store the file as if successful */
        NULL
    };

    curl_global_init(CURL_GLOBAL_DEFAULT);

    curl = curl_easy_init();
    if(curl)
    {
        string url = string_concat((char *[])
        {
            addr, (char *)ftpfile.filename
        }, 2);

        curl_easy_setopt(curl, CURLOPT_URL, url.ptr);

        /* Define our callback to get called when there's data to be
written */
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, my_fwrite);
        /* Set a pointer to our struct to pass to the callback */
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &ftpfile);

        /* Switch on full protocol/debug output */
        curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

        res = curl_easy_perform(curl);
    }
}

```

```

    /* always cleanup */
    curl_easy_cleanup(curl);

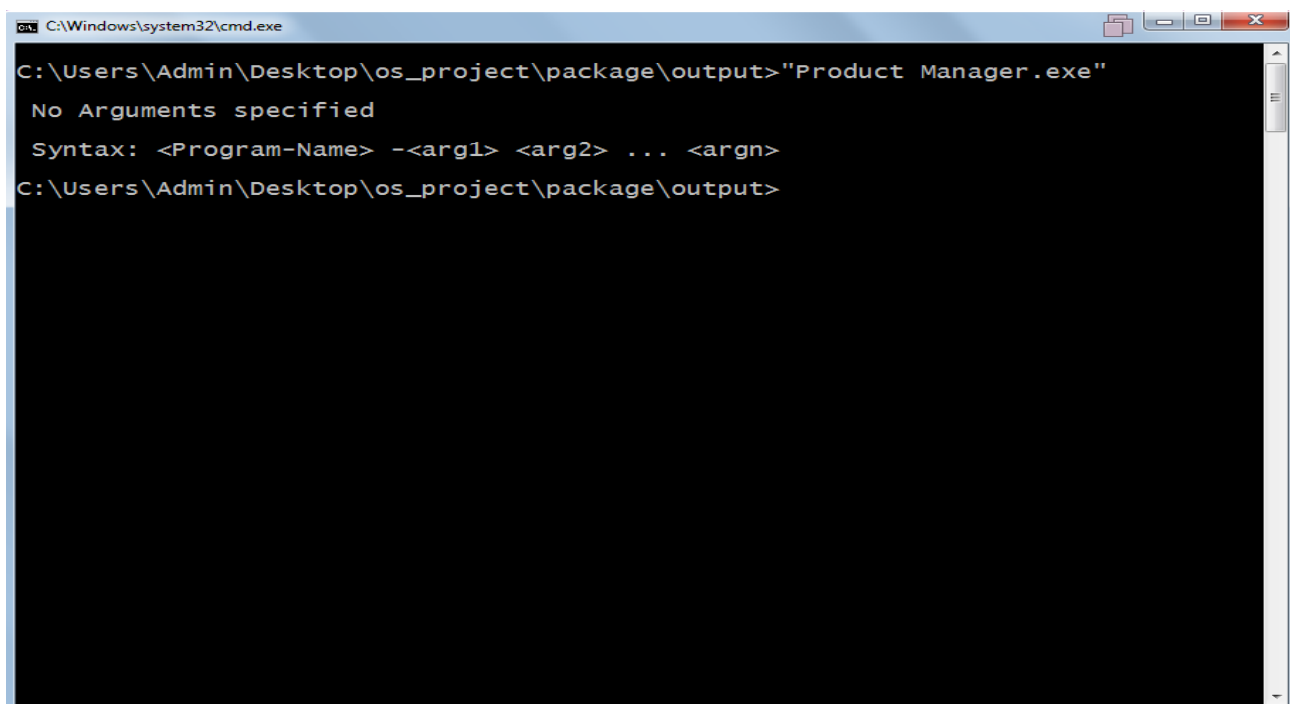
    if(CURLE_OK != res)
    {
        /* we failed */
        fprintf(stderr, "curl told us %d\n", res);
        return 0;
    }
}

if(ftpfile.stream)
    fclose(ftpfile.stream); /* close the local file */
curl_global_cleanup();
return 1;
}
-----END-----

```

In the above code another custom data type is used named struct “FtpFile”. This structure contains the filename and the stream integer which represents the stream the current program instance is using to communicate. The cURL library functions is used to query the server first to verify whether it is functioning properly before downloading. The download is done through the stream that the FtpFile opened.

## 7. Output Screenshots



```

C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe"
No Arguments specified
Syntax: <Program-Name> -<arg1> <arg2> ... <argn>
C:\Users\Admin\Desktop\os_project\package\output>

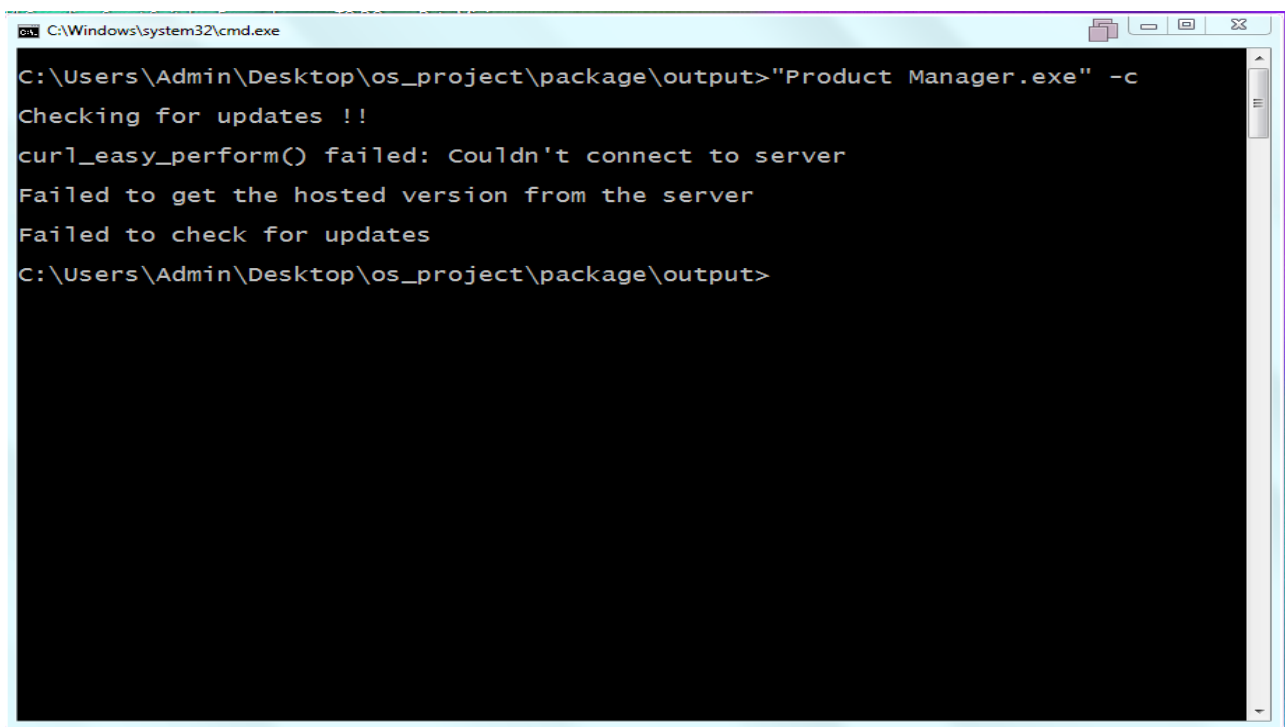
```

Syntax of the Program (Case: When no arguments are specified)



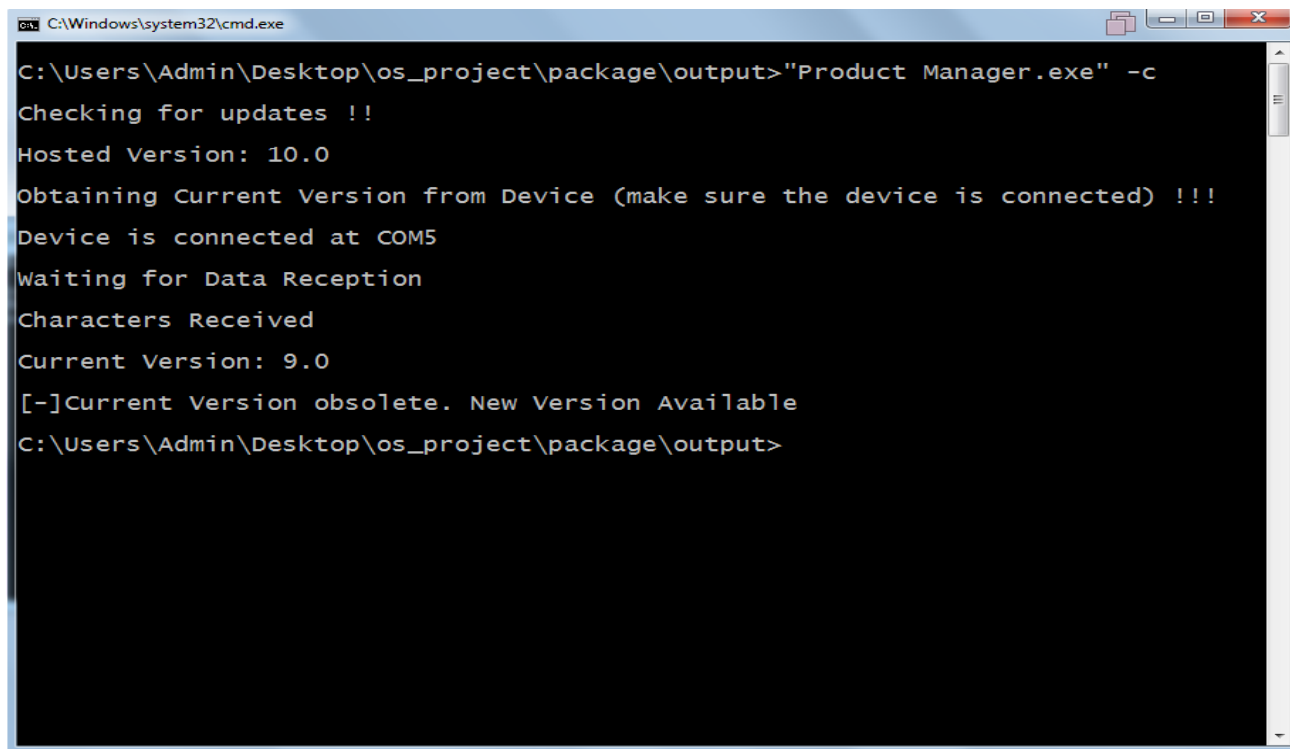
```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -c
Checking for updates !!
Hosted Version: 10.0
Obtaining Current Version from Device (make sure the device is connected) !!!
Device is disconnected
Failed to check for updates
C:\Users\Admin\Desktop\os_project\package\output>
```

Checking for updates (Case: Disconnected Device)



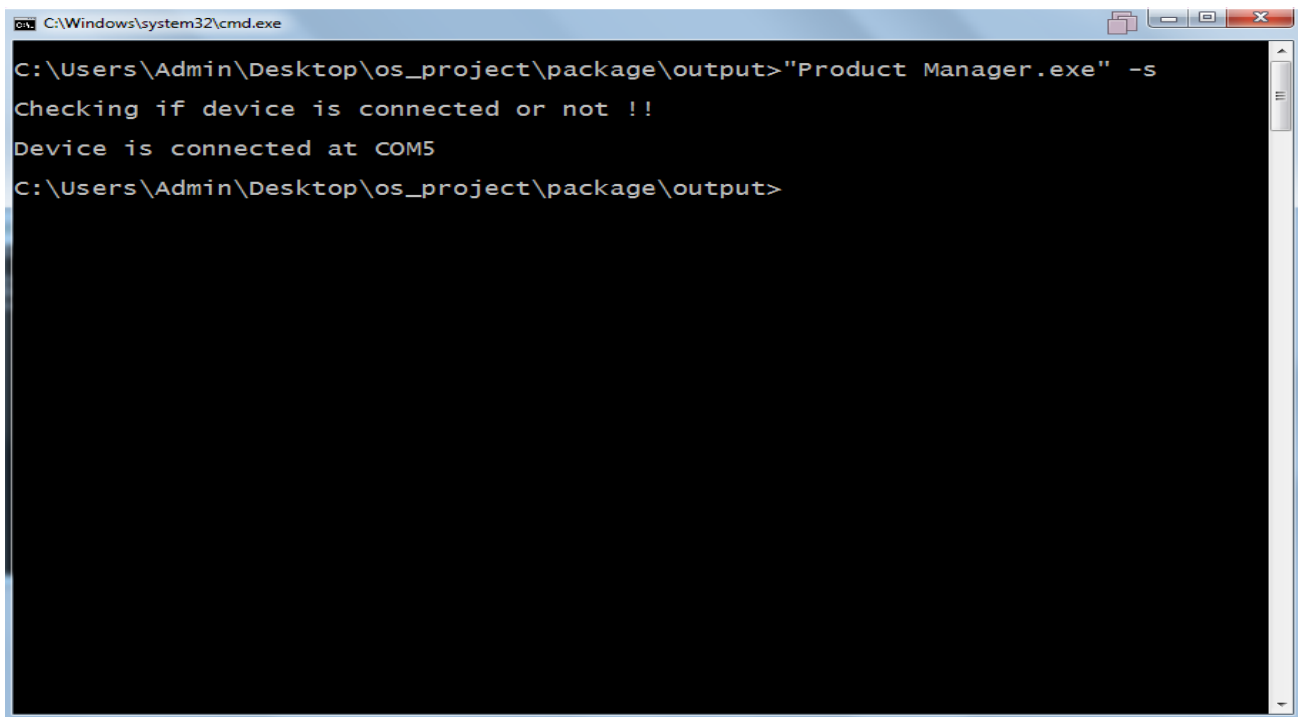
```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -c
Checking for updates !!
curl_easy_perform() failed: Couldn't connect to server
Failed to get the hosted version from the server
Failed to check for updates
C:\Users\Admin\Desktop\os_project\package\output>
```

Checking for updates (Case: Network disconnected)



```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -c
Checking for updates !!
Hosted Version: 10.0
Obtaining Current Version from Device (make sure the device is connected) !!!
Device is connected at COM5
Waiting for Data Reception
Characters Received
Current Version: 9.0
[-]Current Version obsolete. New Version Available
C:\Users\Admin\Desktop\os_project\package\output>
```

Check updates (Case: New version available on the server)



```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -s
Checking if device is connected or not !!
Device is connected at COM5
C:\Users\Admin\Desktop\os_project\package\output>
```

Checking device status (Case: Connected)



```
C:\Windows\system32\cmd.exe

C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -s
Checking if device is connected or not !!
Device is disconnected
C:\Users\Admin\Desktop\os_project\package\output>
```

### Checking device status (Case: Disconnected)

```
C:\Windows\system32\cmd.exe

C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -u
Checking for updates !!
Hosted Version: 10.0
Obtaining Current Version from Device (make sure the device is connected) !!!
Device is connected at COM5
Waiting for Data Reception
Characters Received
Current Version: 9.0
[-]Current Version obsolete. New Version Available
Updating device !!
* Trying 192.168.149.1...
* TCP_NODELAY set
* Connected to 192.168.149.1 (192.168.149.1) port 8081 (#0)
> GET /os_project/package.hex HTTP/1.1
Host: 192.168.149.1:8081
Accept: */*

< HTTP/1.1 200 OK
< Date: Tue, 13 Nov 2018 02:27:23 GMT
< Server: Apache/2.4.23 (win64) PHP/7.0.10
< Last-Modified: Tue, 27 Feb 2018 23:11:15 GMT
< ETag: "a3f-56639be4a194e"
< Accept-Ranges: bytes
< Content-Length: 2623
<
* Connection #0 to host 192.168.149.1 left intact
avrdude: Version 6.3, compiled on Jan 17 2017 at 12:00:53
Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
Copyright (c) 2007-2014 Joerg Wunsch

System wide configuration file is "C:\Users\Admin\Desktop\os_project\package\output\avrdude\avrdude.conf"

Using Port                : COM5
Using Programmer           : arduino
Overriding Baud Rate       : 115200
```

### Updating the device

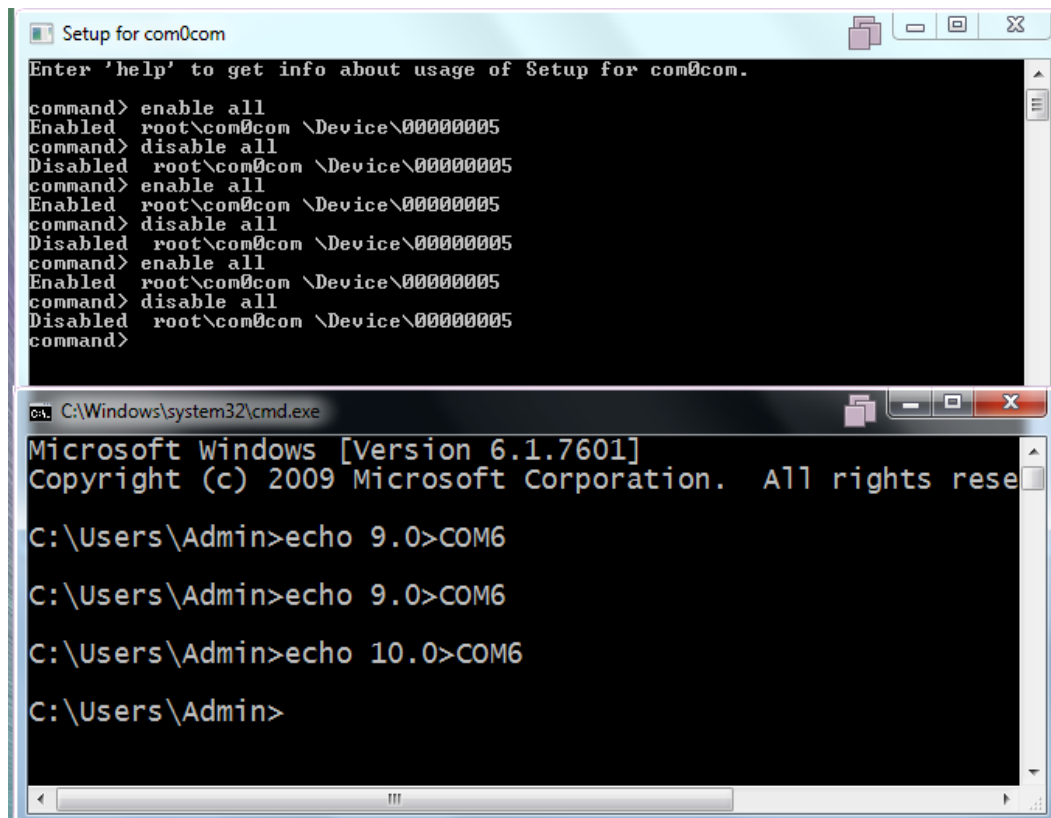
```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -u
Checking for updates !!
Hosted Version: 10.0
Obtaining Current Version from Device (make sure the device is connected) !!!
Device is connected at COM5
Waiting for Data Reception
Characters Received
Current Version: 10.0
[*]Current Version is up to date
C:\Users\Admin\Desktop\os_project\package\output>
```

Checking for updates (Up to date software)

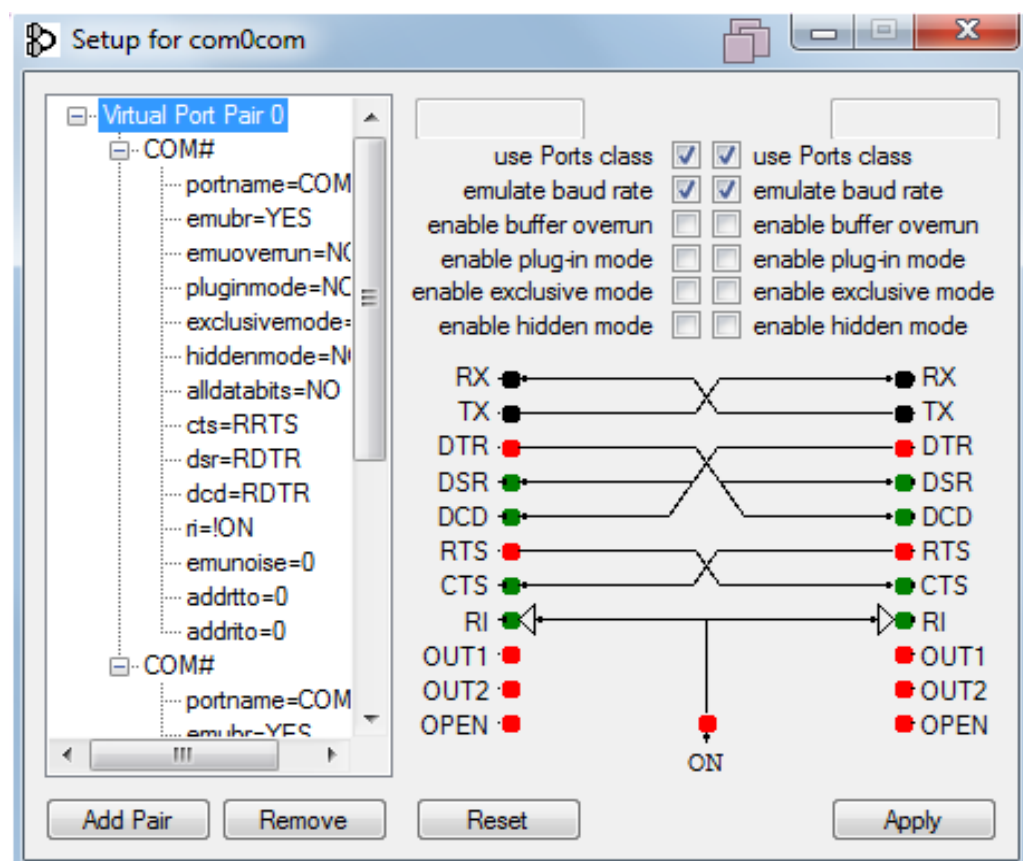
```
C:\Windows\system32\cmd.exe
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe"
No Arguments specified
Syntax: <Program-Name> -<arg1> <arg2> ... <argn>
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -a
Product Manager.exe: invalid option -- a

c-> Check for updates: Connects to server and checks if new version is available
s-> Check for device status: Checks whether the device is connected or not
u-> Upgrade the device: Checks for updates, checks device status and upgrades the device software
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" a
Unrecognized format for specifying arguments
Syntax: <Program-Name> -<arg1> <arg2> ... <argn>
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" csu
Unrecognized format for specifying arguments
Syntax: <Program-Name> -<arg1> <arg2> ... <argn>
C:\Users\Admin\Desktop\os_project\package\output>"Product Manager.exe" -csu
Checking for updates !!
```

Case: Invalid arguments or invalid format of specifying formats



Emulating the connection of a microcontroller



Null com emulator to setup up emulation of virtually linked COM ports

```
C:\Windows\system32\cmd.exe

C:\Users\Admin\Desktop\os_project\package\src>mingw32-make release
cmd /c if not exist bin\\Release md bin\\Release
cmd /c if not exist obj\\Release md obj\\Release
gcc.exe -Wall -DCURL_STATICLIB -O2 -c check_updates.c -o obj\\Release\\check_updates.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c connection_status.c -o obj\\Release\\connection_status.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c ftpget.c -o obj\\Release\\ftpget.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c main.c -o obj\\Release\\main.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c process.c -o obj\\Release\\process.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c str_fnt.c -o obj\\Release\\str_fnt.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c upgrade.c -o obj\\Release\\upgrade.o
gcc.exe -Wall -DCURL_STATICLIB -O2 -c usb2serial_read_w32.c -o obj\\Release\\usb2serial_read_w32.o
g++.exe -o bin\\Release\\Product\\Manager.exe obj\\Release\\check_updates.o obj\\Release\\connection_status.o obj\\Release\\ftpget.o obj\\Release\\main.o obj\\Release\\process.o obj\\Release\\str_fnt.o obj\\Release\\upgrade.o obj\\Release\\usb2serial_read_w32.o -lcurl -lws2_32 -lwldap32 -s

C:\Users\Admin\Desktop\os_project\package\src>
```

Make file used for building the project source

```
C:\Users\Admin\Desktop\os documentation\src>tree /F /A
Folder PATH listing
Volume serial number is 2874-6126
C:.
|   Makefile
|   README.txt
|
+---avrdude
|   avrdude.conf
|   avrdude.exe
|   libusb0.dll
|
+---headers
|   custom.h
|   str.h
|
\---source
    check_updates.c
    connection_status.c
    ftpget.c
    main.c
    process.c
    str_fnt.c
    upgrade.c
    usb2serial_read_w32.c
```

Project file tree

## 8. Results And Discussions

The software is installed using command line and make command. Since this is a tool focused on command line usage, the makefile provides a very good method to manage dependencies and tree of the project. The make file of this project was created using cbp2make tool.

This defines the overall implementation of our project. This entire project makes heavy use of the cURL library. Thus, we have generated the executable with cURL library statically compiled and thus no explicit installation is necessary. Since this program does not involve viewing the server directly, a simple server that only receives and responds to requests can be used to demonstrate the working. The size of the project is thus very small. The project also makes very small but useful use of the PowerShell scripting in Windows.

By implementing this project, we can allow the update delivery of any microcontroller chip to be automated if the user desires it. Also since this is developed in C language, the compiled code is cross platform save for the cURL library.

This tool is currently only supported on Windows. Windows operating system has the most user-friendly experience out of all the other operating systems and a very effective API for C programming that allows manipulation of external devices and drivers connected at that instant. Most of functions used in this project are derived from the “Windows.h” header file for the C programming language. Post installation the “avrdude” folder which is in the same directory is used to update the device.

## 9. Bibliography

1. Serial Communication in Windows:  
<https://www.codeproject.com/Articles/2682/Serial-Communication-in-Windows>
2. C-program for simple serial communication:  
<https://gist.github.com/DavidEGrayson/5e5bb95ae291cdfdffd4>
3. <https://www.codeproject.com/Articles/3061/Creating-a-Serial-communication-on-Win>
4. <https://learn.sparkfun.com/tutorials/terminal-basics/command-line-windows-mac-linux>
5. <https://docs.microsoft.com/en-us/dotnet/api/system.io.ports.serialport>
6. popen references:  
[http://www.cs.uleth.ca/~holzmann/C/system/shell\\_commands.html](http://www.cs.uleth.ca/~holzmann/C/system/shell_commands.html)
7. Technical Paper:  
<https://web.archive.org/web/20120905123232/http://www.robbayer.com/files/serial-win.pdf>
8. <https://msdn.microsoft.com/en-us/library/ff802693.aspx>
9. Static Building of curl:  
<https://fatchoco.wordpress.com/2009/03/01/using-curl-in-mingw>