ECE 540 FINAL PROJECT : VIRTUAL RUBIK’S CUBE

PROJECT REPORT

TEAM:
GUNADEEP REDDY BADE
SAI CHAITANYA VARMA KALIDINDI
HARANADH CHINTAPALLI
SHRUTHI NARAYAN REDDY

Project Description

In this project, we are Implementing 3X3 Virtual Rubik’s cube puzzle. This game is displayed on a VGA monitor interfaced with Nexys 4 board. In the puzzle, the Cube should be of solvable configuration and start from the random position. The Cube has to be solved can be moving the cube in required direction out of 18 possible cube movements. Blynk API is used to provide user interface to select these movements. These signals are captured using ESP8266 and sent to Nexys4DDR board over PMOD Port. Cube logic is implemented in Firmware which decodes the movement and updates the cube configuration registers. These registers are used by colorizer module to update cube configuration

Rubik’s Cube

Rubik’s Cube is a 3-D combination puzzle invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik. On a classic Rubik's Cube, each of the six faces is covered by nine stickers, each of one of six solid colors: white, red, blue, orange, green, and yellow. In currently sold models, white is opposite yellow, blue is opposite green, and orange is opposite red, and the red, white and blue are arranged in that order in a clockwise arrangement. On early cubes, the position of the colors varied from cube to cube. An internal pivot mechanism enables each face to turn independently, thus mixing up the colors. For the puzzle to be solved, each face must be returned to have only one color. Following are the possible Cube rotations. In our project we have included 6 more rotations to rotate cube clockwise and anti-clockwise in X,Y and Z axis.

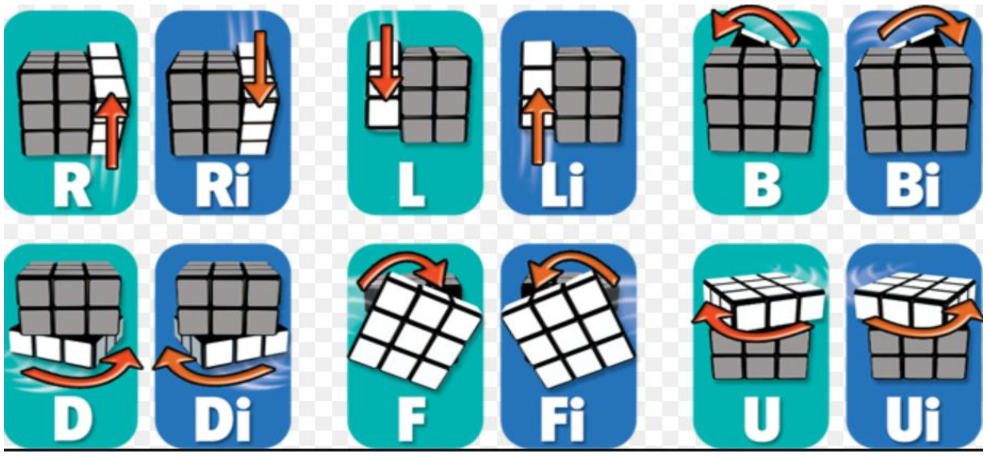


Fig.1 Cube Rotations

Hardware and Software Requirements:
Physical Hardware: Nexys4DDR Xilinx FPGA,ESP8266,Breadboard,Leds ,Connecting wires
Firmware: MIPS (Mips assembly)
HDL used: Verilog

Block Diagram:

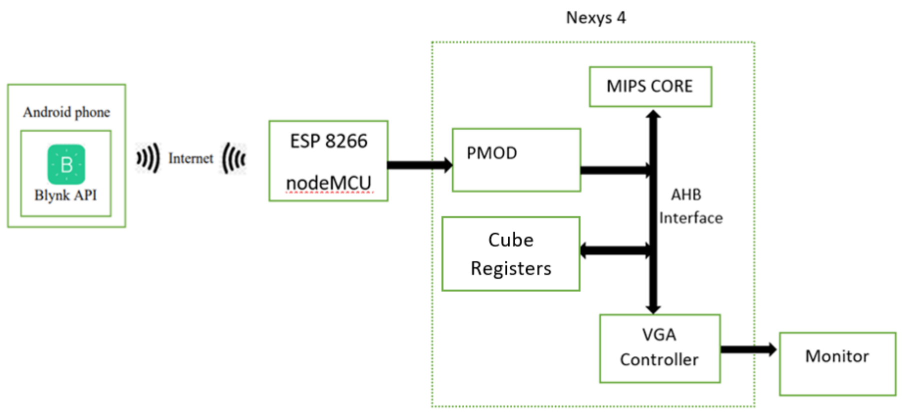


Fig 2. High level Block diagram

The following features from the Nexys4 DDR Board have been used by us in this Project

- PMOD- to receive data from nodeMCU
- VGA controller – Used for display
- Push-button – Used to Start and Reset Game

PMOD Ports

The Pmod ports are arranged in a 2×6 right-angle.Each 12-pin Pmod port provides two 3.3V VCC signals (pins 6 and 12), two Ground signals (pins 5 and 11), and eight logic signals, Port JB has been used by us for this project

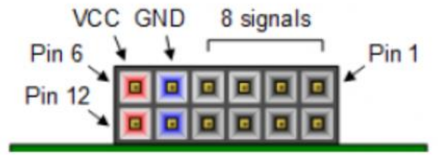


Fig 3. PMOD Port interface

VGA Controller:

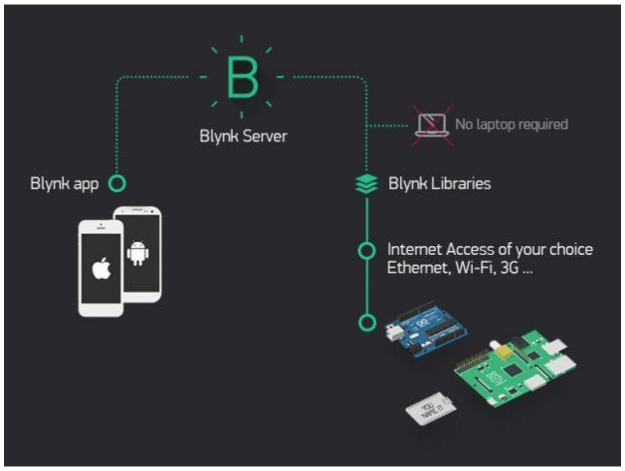
Video Graphics Array or VGA is a display standard which provides 640 X 480 standard resolution color display screens. The Nexys4 DDR board has VGA port with 4 bits per color and two standard sync signals for horizontal sync and vertical sync to be interfaced with the VGA display monitor. We have designed our game such that it initially displays "Press Start" image on VGA. On Pressing the any push Button game will be started and the VGA screen consists Cube. Upon successfully solving puzzle Congratulations will be displayed on the screen.

Push Buttons:

Nexys4 DDR board has six pushbuttons as a part of basic I/O. Among those the Red push button is a "CPU RESET" button. In this project we used two of the six push buttons. We have used the CPU Reset button to reset our game and any of the five buttons, to trigger the start of game. Pushbutton is also used to select random initial configuration.

Blynk:

Blynk is a Platform with iOS and Android apps to control Arduino, Raspberry Pi which was designed for the Internet of Things. It can control hardware remotely, display sensor data, it can store data, visualize it It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets. Blynk is a smartphone application that allows you to easily create "apps" that interact with Internet-connected hardware. It works with a wide variety of hardware platforms, including the Photon, Raspberry Pi, Arduino/Ethernet Shield, and, of course, the ESP8266. We have used ESP8266 and then interfaced this to PMOD's JB port. Refer to Appendix for steps to create Blynk app. Blync Project code used to encode 18 movements into 5 bit from 1 to 15 is included in project folder.



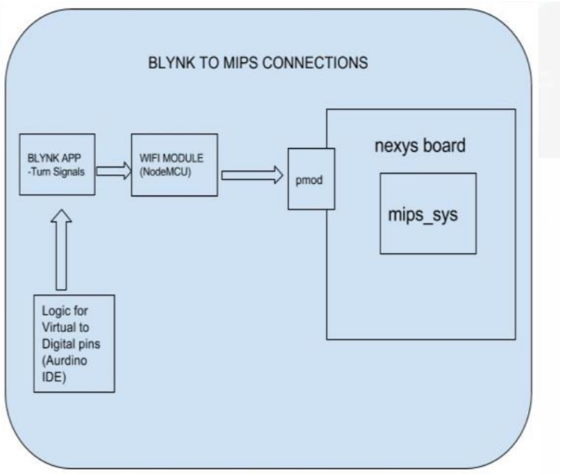

BLYNK TO MIPS CONNECTIONS
BLYNK APP -Turn Signals
WIFI MODULE (NodeMCU)
pmod
nexys board
mips_sys
Logic for Virtual to Digital pins (Aurdino IDE)

Fig 4. Blynk API operation Connectivity

BLYNK APP SCREENSHOT:

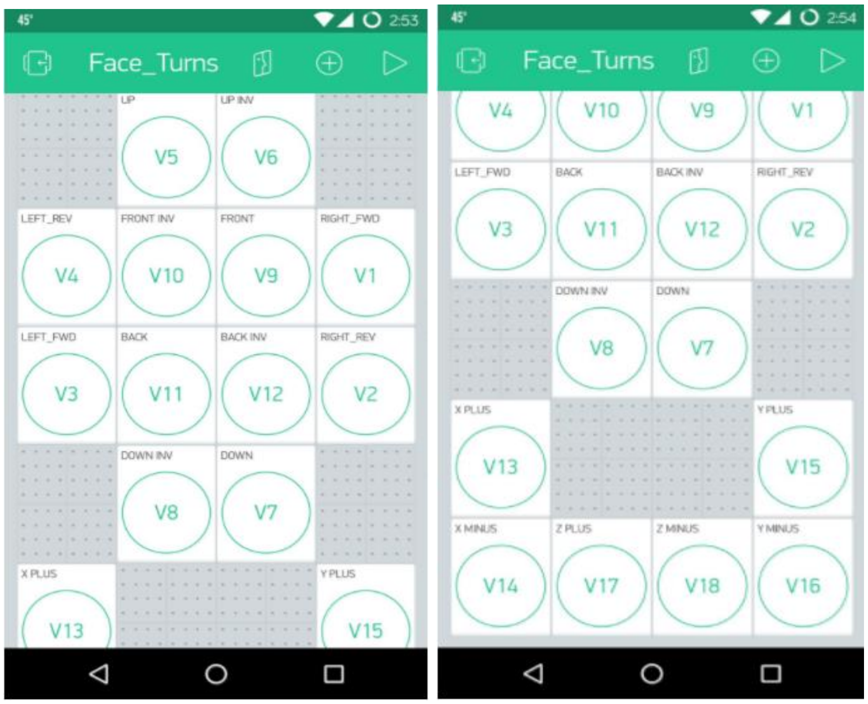


Fig 5. Screenshot of Blynk app with rotations

CUBE Pheripheral

Designed new peripheral for the Cube registers IO_FACEFRONT, IO_FACERIGHT, IO_ FACEUP, IO_FACEDOWN , IO_FACEBACK,IO_FACELEFT.IO_START is used to indicate that cube solving has started. MIPS Firmware updates IO_DONE=1 to indicate that cube puzzle has been solved. Each cube face is divided into 9 Facets .Each facet needs 3 bit to encode colors. 2 is decoded as Blue.3 is decoded as GREEN,4 is decoded as WHITE,5 as ORANGE,6 as RED,7 as YELLOW. Default as PINK. Bits 0 to 26 is used to encode colors for facets in each FACE registers. Bits 27 to 31 are considered as don't cares

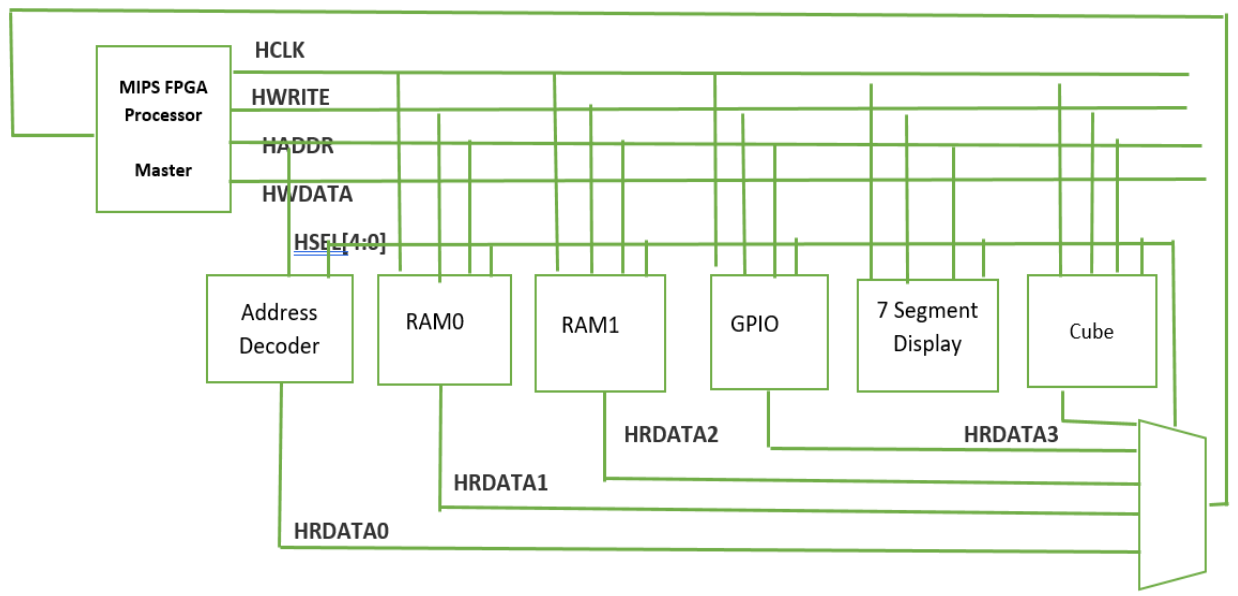


FIG 6.AHB Peripheral

CUBE Registers:

The MIPSfgpa system in the Getting Started project implemented three address ranges (RAM0, RAM1 and GPIO).RAM0 is 1 KB and holds the boot code (virtual addresses 0xbfc00000-0xbfc003fc = physical addresses 0x1fc00000-0x1fc003fc). RAM1 is 256 KB and holds the user code (virtual addresses 0x80000000-0x8003fffc = physical addresses 0x00000000-0x0003fffc). The LEDs, switches, pushbuttons, and your seven segment display registers are mapped to virtual memory addresses as shown in Table 1. The processor code uses virtual memory addresses, and the AHB-Lite bus receives physical addresses.

Memory Addresses for Nexys4 DDR FPGA board:

Virtual Address	Physical Address	Signal Name	Nexys DDR
0xBF70_0000 - 0xBF70_000C	0x1F70_0000 – 0x1F70_000C	IO_7SEGEN_N (digit enables), IO_SEG_N (segment values)	Seven segment display
0xBF80_0000	0x1F80_0000	IO_LED	LEDs
0xBF80_0004	0x1F80_0004	IO_SW	Switches

0xBF80_0008	0x1F80_0008	IO_PB	U, D, L, R, C pushbuttons
0xBF80_0018	0x1F80_0018	IO_START	Start
0xBF80_0040	0x1F80_0040	IO_DONE	Done
0xBF80_0038	0x1F80_0040	IO_JB	PMOD for BlynK Signals
0xBf800020- 0xBf800034	0x1f800020- 0x1f800034	IO_FACEFRONT, IO_FACERIGHT, IO_ FACEUP, IO_FACEDOWN, IO_FACEBACK, IO_FACELEFT	Cube Register Front, Cube Register Right, Cube Register Up, Cube Register Down, Cube Register Back, Cube Register Left

Firmware Logic : (MIPS Assembly)

Code is included in the root directory.

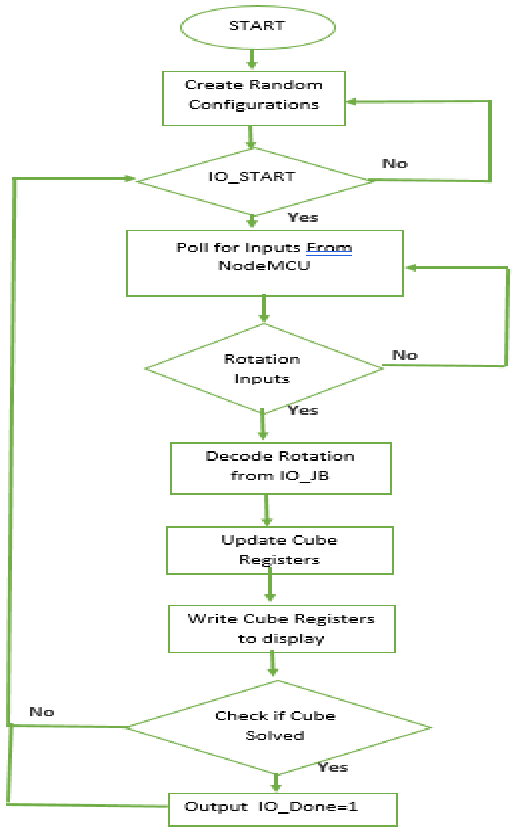
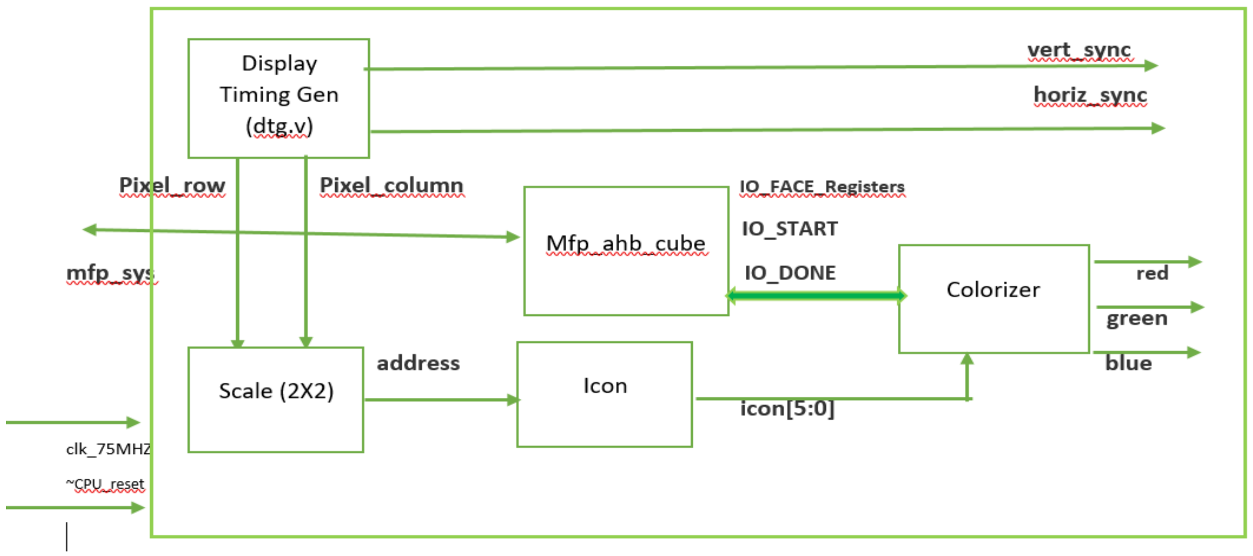

START
Create Random Configurations
IO_START
No
Yes
Poll for Inputs From NodeMCU
Rotation Inputs
No
Yes
Decode Rotation from IO_JB
Update Cube Registers
Write Cube Registers to display
Check if Cube Solved
No
Yes
Output IO_Done=1

Fig 7. Flow chart for Firmware

Display Sub System:



A high-level block diagram for the display subsystem is shown above. We have used a model similar to that of Rojobot provided to us in Lab 2 with an exception that no world map is used .We have used a single map where COE file consists of previously encoded value for each facets. Upon receiving the encoded value we are deciding on the color to be displayed based on the decoded value for that particular facet. The subsystem uses a bit map to produce the image on the display. The Display Timing Generator (dtg.v) produces the pixel row and pixel column addresses. The 512 x 384 world is displayed as a 1024 x 768 image on the VGA monitor so the implementation scales to represent each world location as an 2 x 2 pixel block on the display. The resulting pixel stream is sent to the Colorizer to draw an image of the map. The colorizer module decodes the value of face Registers

Display Timing Generator

The Display Timing Generator (DTG) generates the video raster signals Vertical Sync (vert_sync), Horizontal Sync (horiz_sync), and video_on which indicates the viewable region of the video screen; and pixel_row and pixel_column, which indicate the current vertical and horizontal pixel position on the display. Refer to Nexys4 DDR Reference Manual for details.

The display image has 768 rows with 1024 pixels (columns) in each row. Each pixel consists of three colors: Red, Green, and Blue. The Nexys4 DDR supports 4096 colors (4 color bits per pixel for red, green, and blue for a total of 12-bit color). The pixel information is delivered to the monitor serially by row. The end of each row is indicated by a *horizontal sync pulse* of a particular length. At the end of all 768 rows, a longer, *vertical sync pulse* occurs. At the end of a horizontal sync pulse, the beam goes back to pixel 0 of the next row. At the end of a vertical sync pulse, the beam goes to pixel {0, 0} (the top left pixel on the display). The entire image is drawn on the display 60 times per second.

Project Tasks

- Implement Cube Logic in Mips Assembly

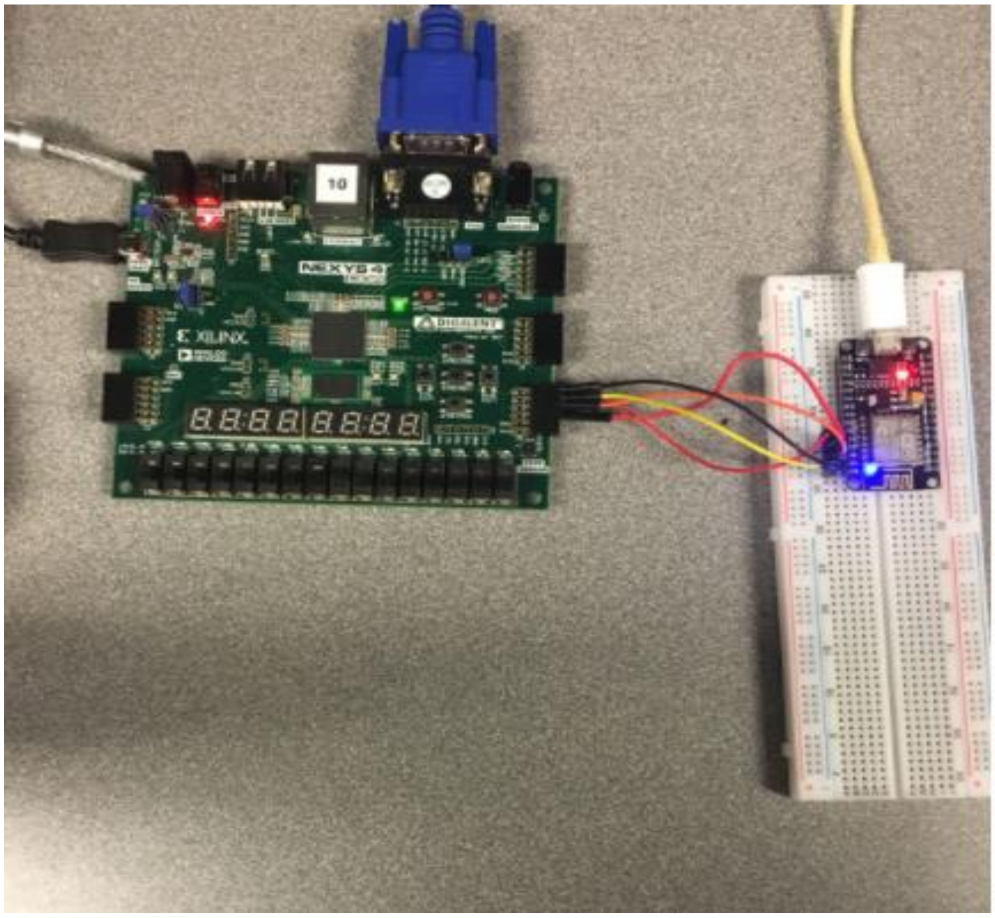
- Create Blynk project to add Buttons for rotation
- Integrate ESP8266 with Nexys4DDR board
- Design and implement the Colorizer module.
- Generate and integrate the 75MHz clock using the IP Integrator Clocking Wizard.
- Integrate the Display Timing Generator, Icon, and Colorizer modules into your system.
- Design mfp_ahb_cube pheripheral
- Add VGA signal output ports to mfp_nexys4_ddr.v and uncomment the VGA connector signals and JB signals in the constraint file (mfp_nexys4_ddr.xdc)

Stretch Goals Completed:

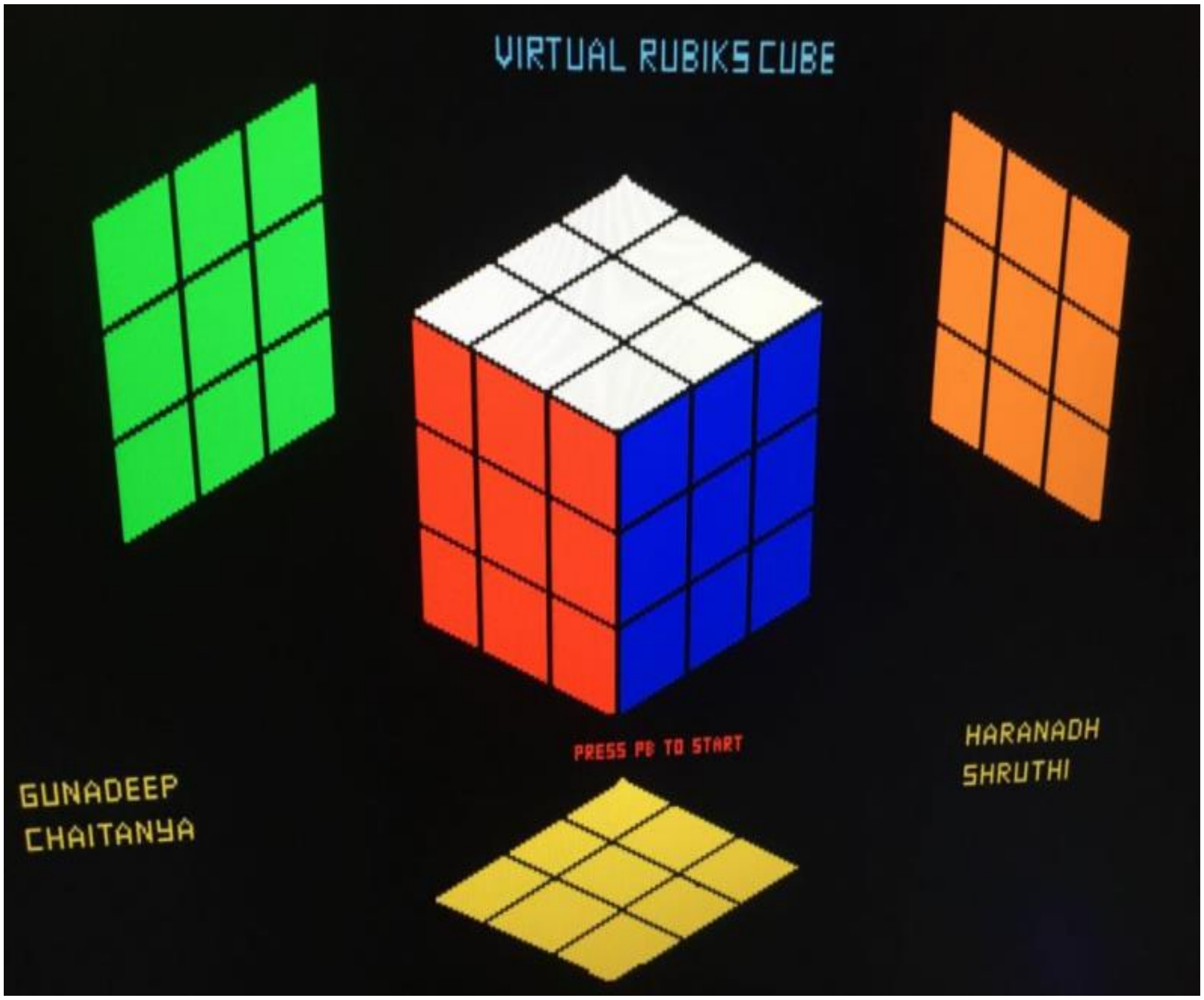
- Interface BLYNK application
- Display Cube as proposed instead of lateral 2D image

Screenshots:

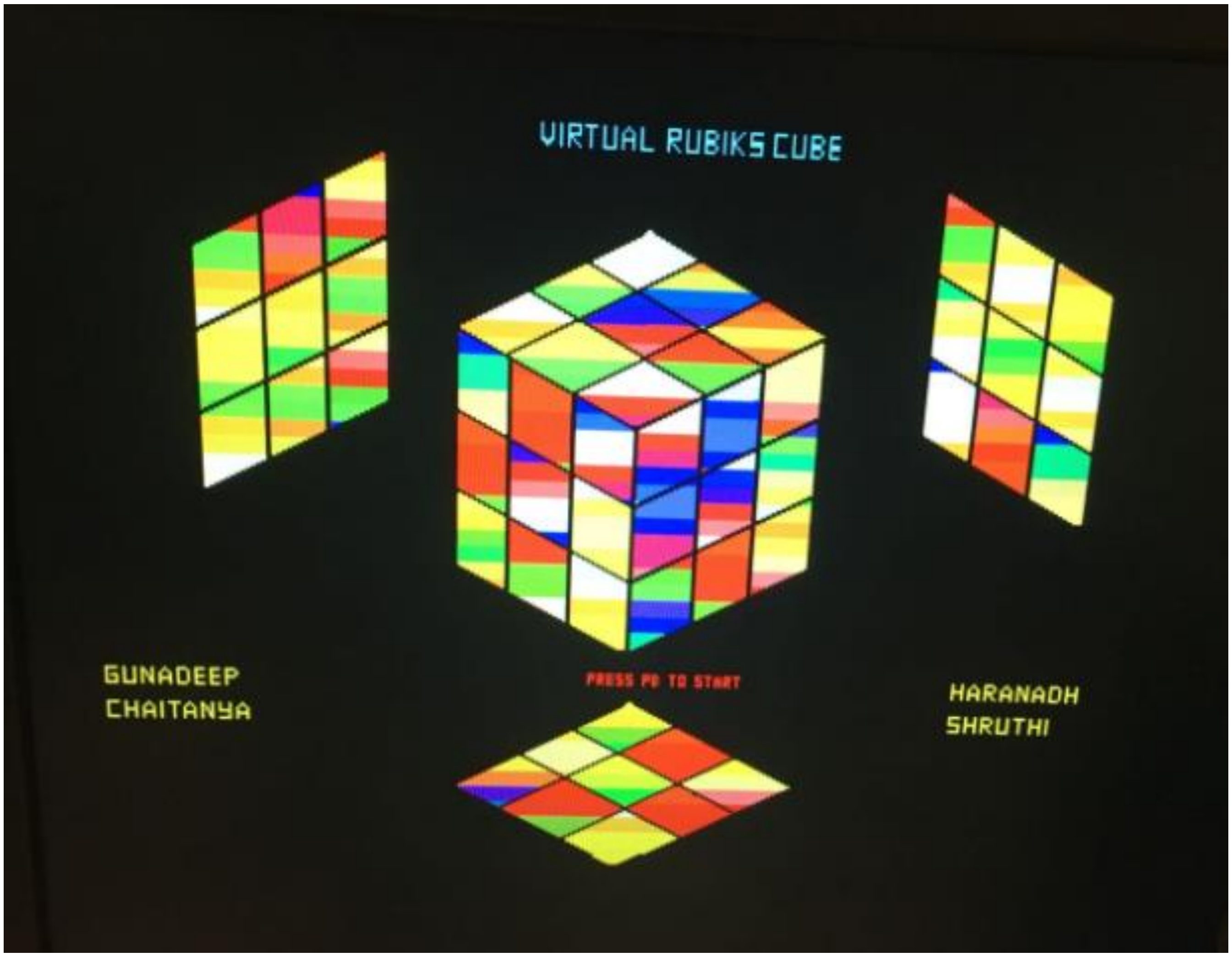
Connections To ESP8255 and Nexys board



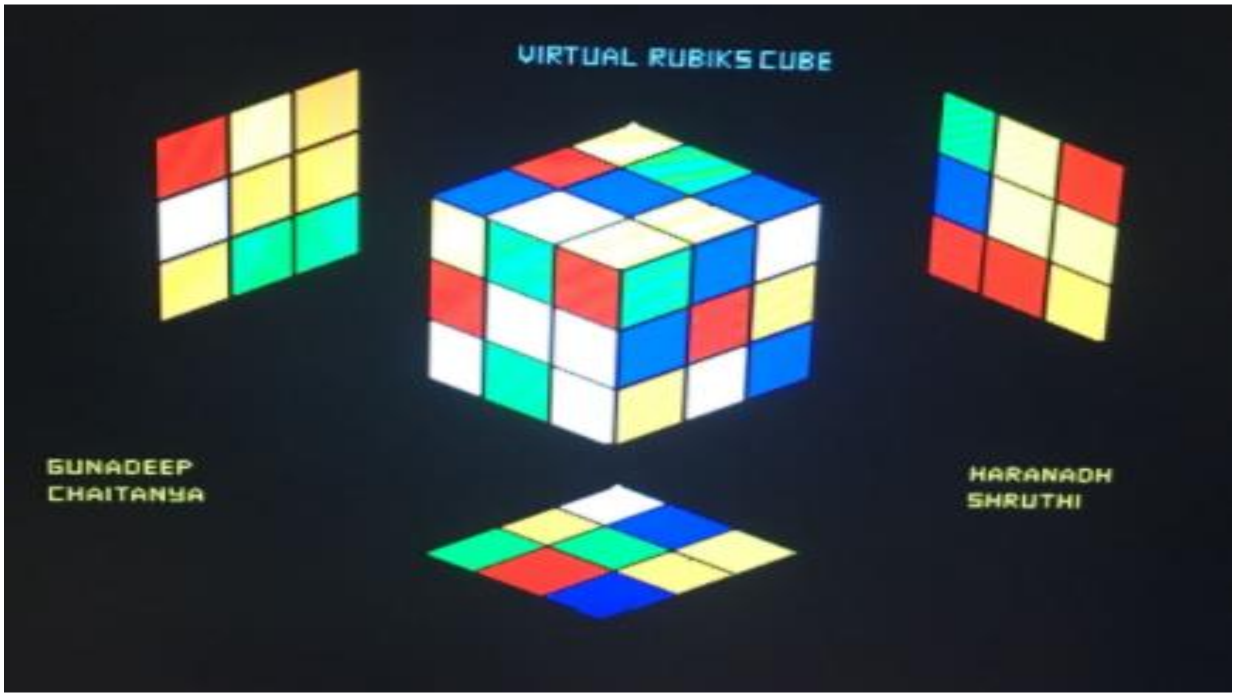
Upon Reset to start Game



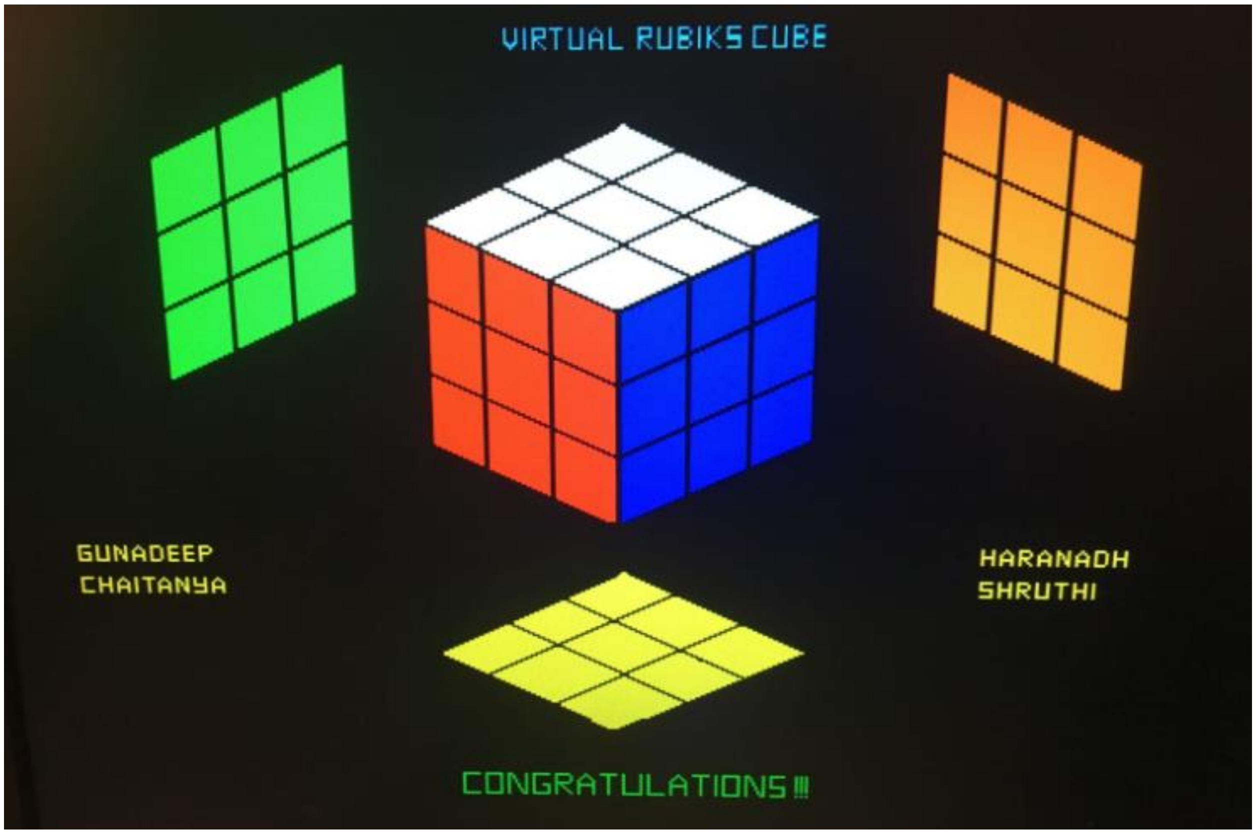
Randomize Initial Configuration



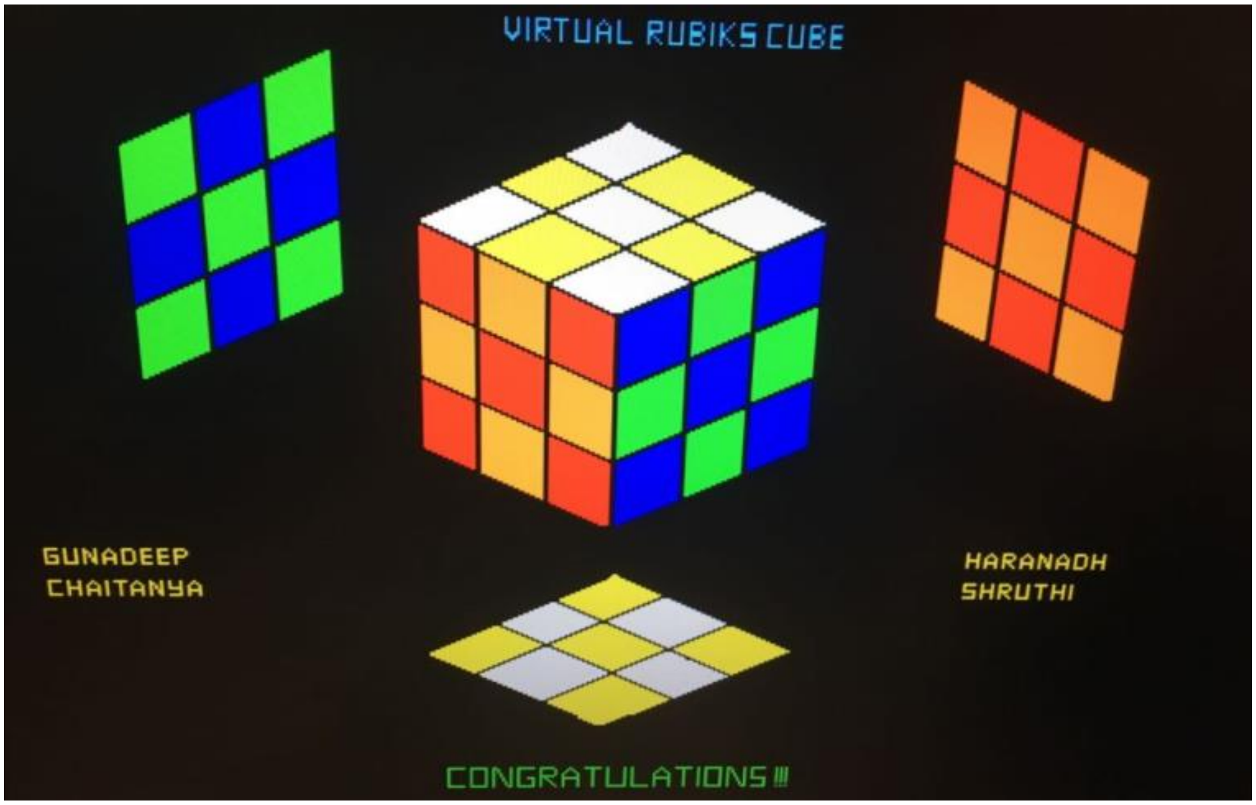
Starting Screen for Puzzle



Congratulations Upon Puzzle Solving



User can still play with the cube after solving it once. Can create patterns



References

Prof.Roy Kravitz Lecture notes and Project-2 manuals

https://en.wikipedia.org/wiki/Rubik's_Cube

http://www.instructables.com/id/Simple-Led-Control-With-Blynk-and-NodeMCU-Esp8266-/

https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual

https://thecube.guru/online-3d-rubiks-cube/

https://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/s2015/akw62_rq35_sp2283/akw62_rq35_sp2283/index.html#thumb

Appendix :

1.Get the App and Arduino Library

The Blynk app is available for both iOS and Android devices. Download app from playstore or Appstore. After downloading the app, create an account and log in. Welcome to Blynk!

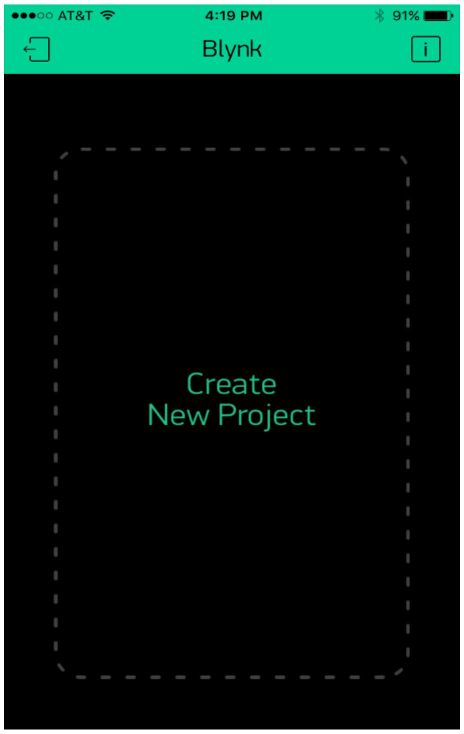


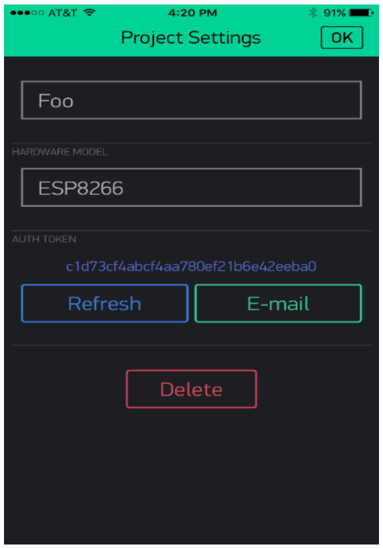
Fig1. Welcome screen

2. You'll also need to install the Blynk Arduino Library, which helps generate the firmware running on your ESP8266. Download the latest release from Blynk's GitHub repo, and follow along with the directions there to install the required libraries.

https://github.com/blynkkk/blynk-library/releases

3. Create a Blynk Project

Next, click the "Create New Project" in the app to create a new Blynk app. Give it any name you please, just make sure the "Hardware Model" is set to ESP8266. The Auth Token is very important – you'll need to stick it into your ESP8266's firmware. For now, copy it down or use the "E-mail" button to send it to yourself.



Foo

HARDWARE MODEL

ESP8266

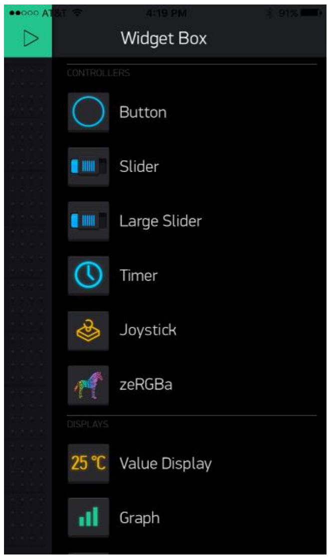
AUTH TOKEN

c1d73cf4abcf4aa780ef21b6e42eeba0

Fig2. Create Project

Add Widgets to the Project

Then you'll be presented with a blank new project. To open the widget box, click in the project window to open.



Add a Button, then click on it to change its settings. Buttons can toggle outputs on the ESP8266. Set the button's output to gp5, which is tied to an LED on the Thing Dev Board. You may also want to change the action to "Switch."