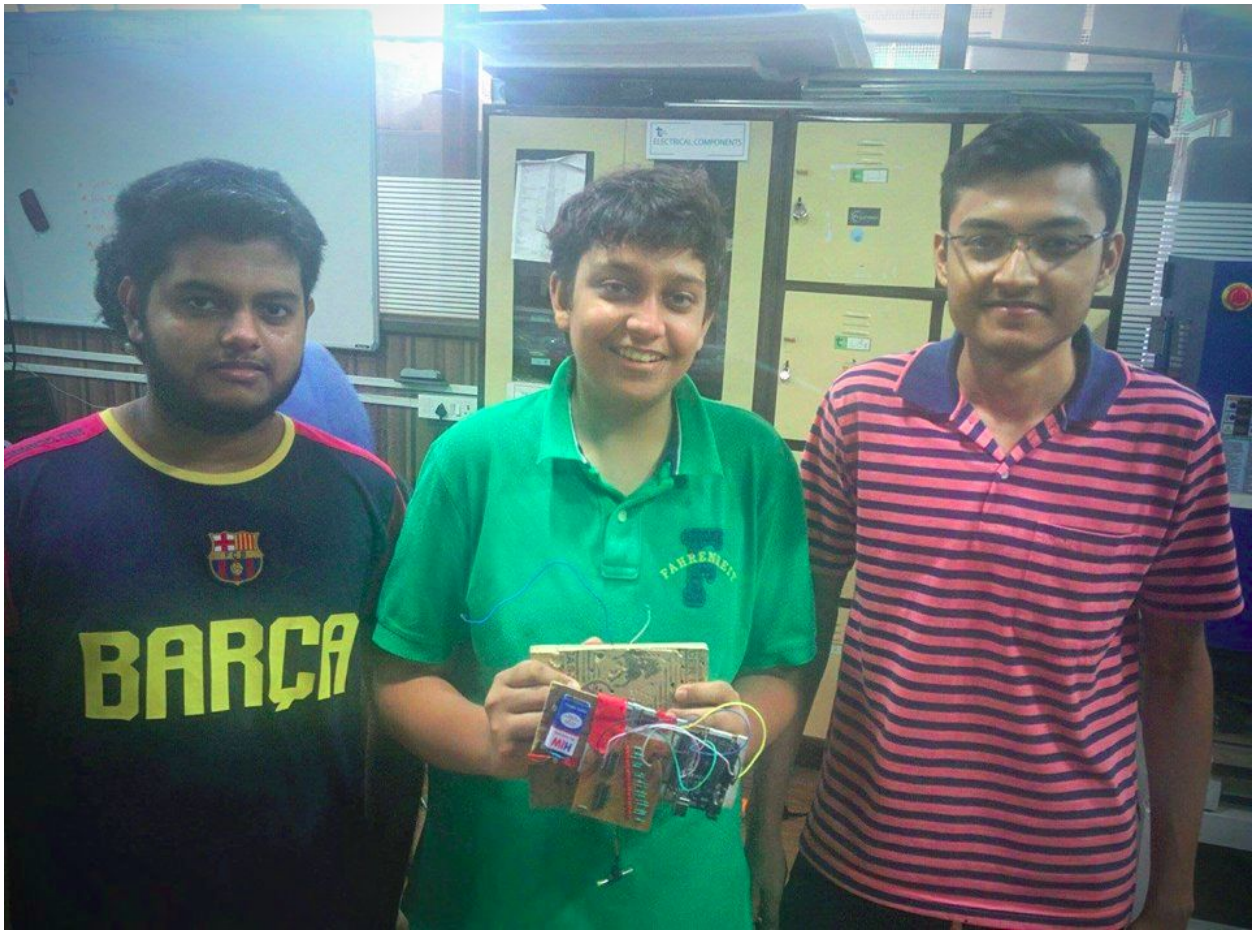


RoLED Display

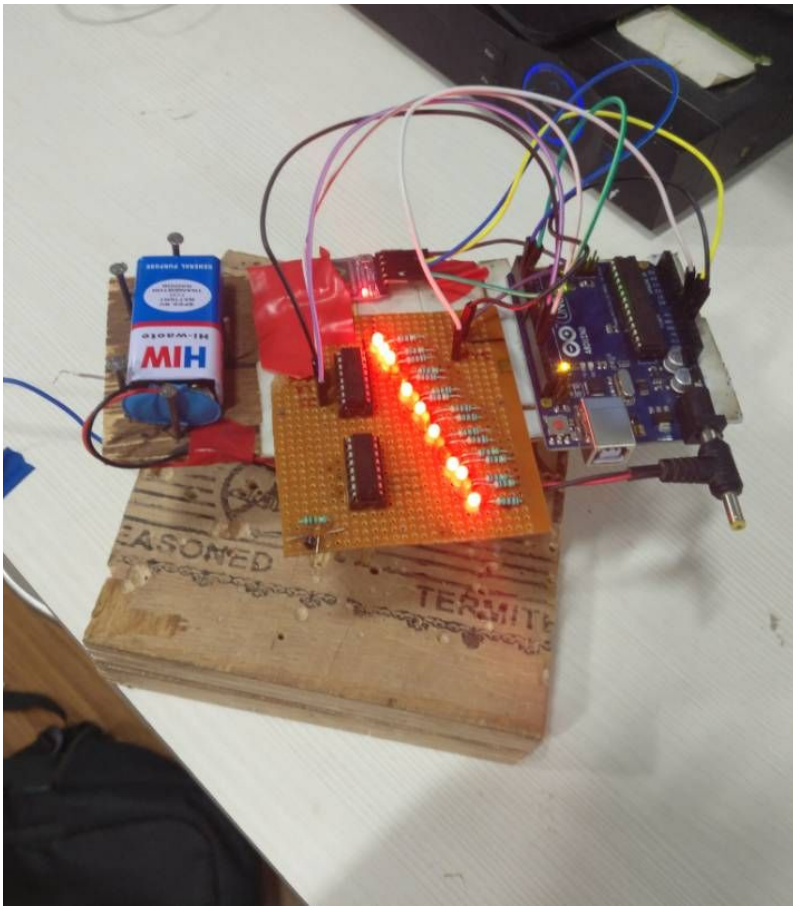
02.06.2017

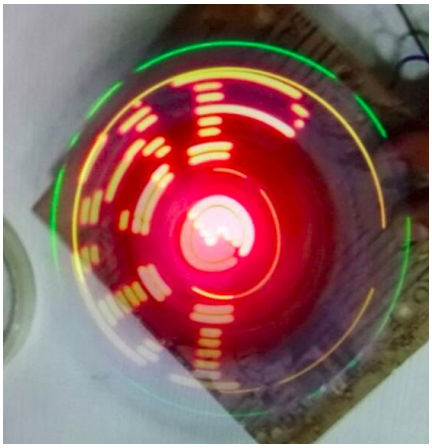
ElectroCoders
Yash Shah (L), Beni Madhav Agarwal, Sarthak Mallick
IIT Bombay

Brief description

We set upon the task to build a circular display (by rotating an array of 16 LEDs at high RPM), which would use persistence of vision to display static images and text, on which we could build applications. Our primary motivation was to drastically reduce the number of LEDs needed to display an image. As of now we can display any image given as input (with reasonable accuracy, due to resolution constraints), as well as words (maximum 20 characters).

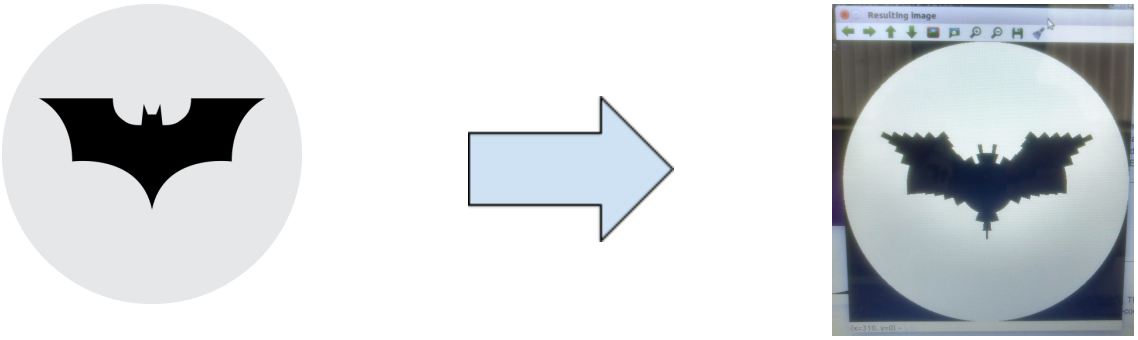
Have a look at our final project here:
https://youtu.be/RErxVAd2anQ
The source code for the same has been shared here:
https://github.com/ys1998/RoLED-Display





About the Project

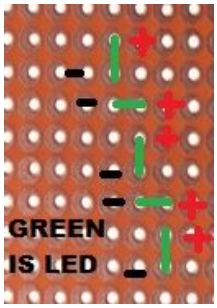
We fixed Arduino, LEDs and Bluetooth module on a PCB (mounted on a wheel for stability) which is rotated by a motor. To display any image, it is first processed to give a circularly cropped image that has been imaginarily divided by concentric circles and sectors into regions (i.e. image is radially pixelated) to give an image that can be displayed by RoLED display:



The data is then sent to the Arduino through the Bluetooth module. The Arduino controls the flicker rate of the LEDs and displays the image.

Technical aspects

Electronic design



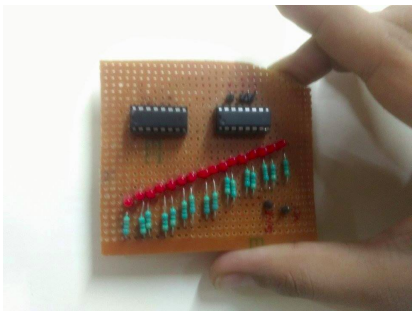
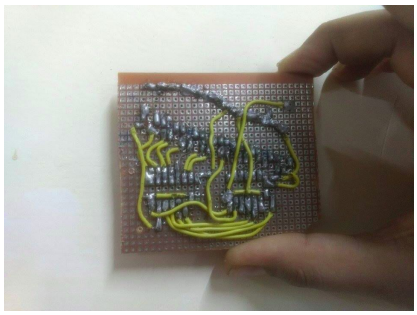
- **Making the led array:** The line of leds had to be as close as possible so that the photo is as continuous as possible.So we reduced the size of LED from 5mm to 3mm. Even after such reduction, there was a gap of 1 hole between 2 LEDs. So the method (shown right) was used. After many trials and errors, this was the closest the LEDs could get.

- **Power supply to rotating circuit:**
  - ***Initial Plan :*** Use the shaft as one terminal (Ground) ,since it is internally shorted to the metal ring around the motor(Insert image). For the second terminal ,aluminium foil was wound around the wheel and a metal plate was attached to the stationary part of the motor , which was pressed so that it made contact with the foil.
  - ***Problem Faced :*** The contact created a lot of friction and the circuit couldn't rotate as fast as required.Tried to reduce friction by using a stern wire in place of metal sheet. But the contact was not perfect , hence initial plan failed. Another solution that we tried was wireless transmission of power using transformer concept; but it provided AC power and arduino needed DC , furthermore switching to AC to DC was an issue. The final solution was way simpler and efficient.
  - ***Final Solution:*** Though we didn't want an onboard power supply due to danger of battery flying at high speed and increasing the weight of the rotating part of the project, we adopted it nevertheless since it was the only possible solution.

- **Attaching the Circuit to the wooden board:** The attachment had to be sturdy and fail proof, since the board would be rotating at high speed.
  - ***Arduino:*** Made holes in the wood and attached it with nuts and bolts through the holes provided in the arduino.Also padded the bottom with DST to provide cushioning, insulation and extra gripping to the arduino.
  - ***Battery:*** The solution wasn't easy to find: we needed a contraption that was strong yet made it easy to replace the 9V battery since batteries needed to be replaced. So we made a boundary on the board where the battery was to be attached and drove nails on the sides of the battery through the wood, leaving a gap of 2-4 mm between the battery and the nails. This made battery changing

easy, since you could just rotate the battery by 90 Degrees and it would come out easily.

- ***The PCB (circuit board):*** The first led had to be the center of revolution and the PCB had to be fixed perfectly. It was fixed by enlarging some of the PCB holes just a bit so that a thin nail could pass through them. Then the PCB was padded with DST to provide cushioning, insulation, and extra gripping to the PCB.

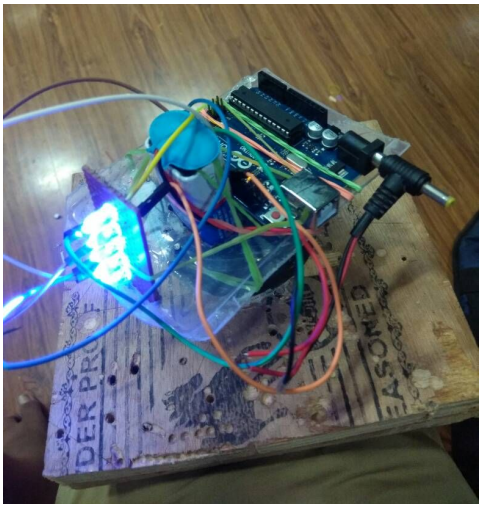



- **Controlling Multiple LEDs with limited number of pins on Arduino:**
  - ***Solution:*** Using one shift register we can control 8 LEDs ,and then connecting them in series one can theoretically control infinite LEDs

- **Damping the vibrations of the motor:** Since there is circuitry above the motor and it rotates at a high rpm, the motor vibrates heavily. Damping was achieved by cutting a hole in a block of wood just right to fit the motor and then filling the gaps using glue gun.

Version 1.0:

This version was a rectangular display instead of the current radial one.
It was a very crude version built on plastic plate with components attached with DST,tape and rubberbands.



Prototype in action:

- Prototype display :

  https://youtu.be/TL0_EPqhmVA

- Working demonstration (see if you can make out “HELLO WORLD”) :

  https://youtu.be/RwzzDwxhLKs

Programming

The Arduino is fed with a sketch before any image data is uploaded. This sketch has the code for receiving data via bluetooth, processing it, and storing it in an array in the desired format. It also has the code to read from this data array, extract the LEDs' state from it and control the flickering of LEDs based on this data and time-delay.

Overall process

- Console application is initiated by executing the bash script `roled.sh`
- `Static image` or `Text` mode is selected
- In case of image mode, the image to be displayed is chosen by providing the complete path to it; in text mode, the text to be displayed is entered
- Corresponding image is generated and previewed
- Previewed image is processed using OpenCV and the image-data (in the form of state of LEDs at a particular instant) is stored in a text file (sector wise); the processed image is then displayed
- Python script runs and transfers data (one character at a time) from the text file to Arduino via bluetooth
- Bluetooth module receives the data and stores it in the specified format in SRAM
- Displaying of image starts once data reading is complete

Image processing algorithm for 'text' mode

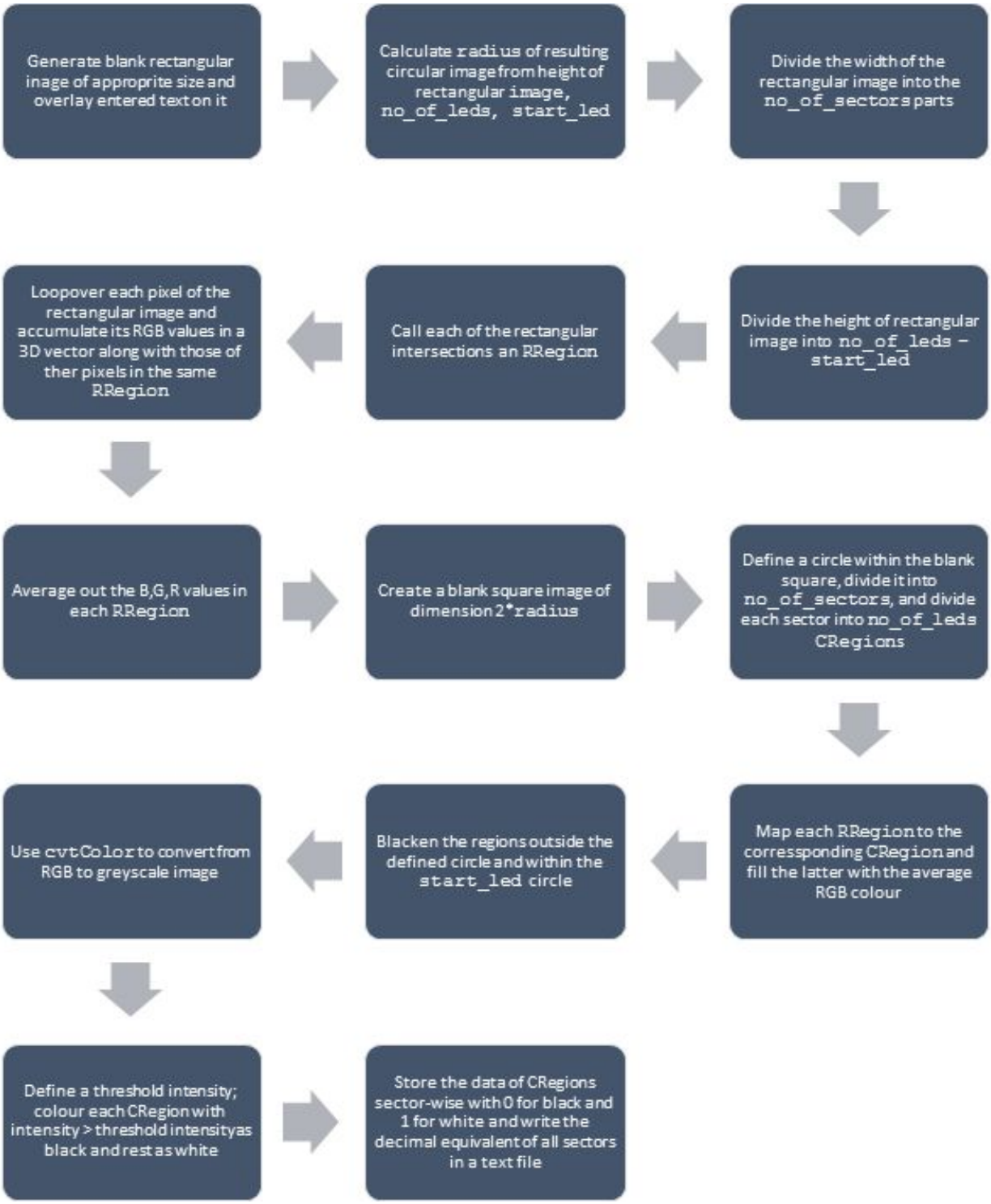
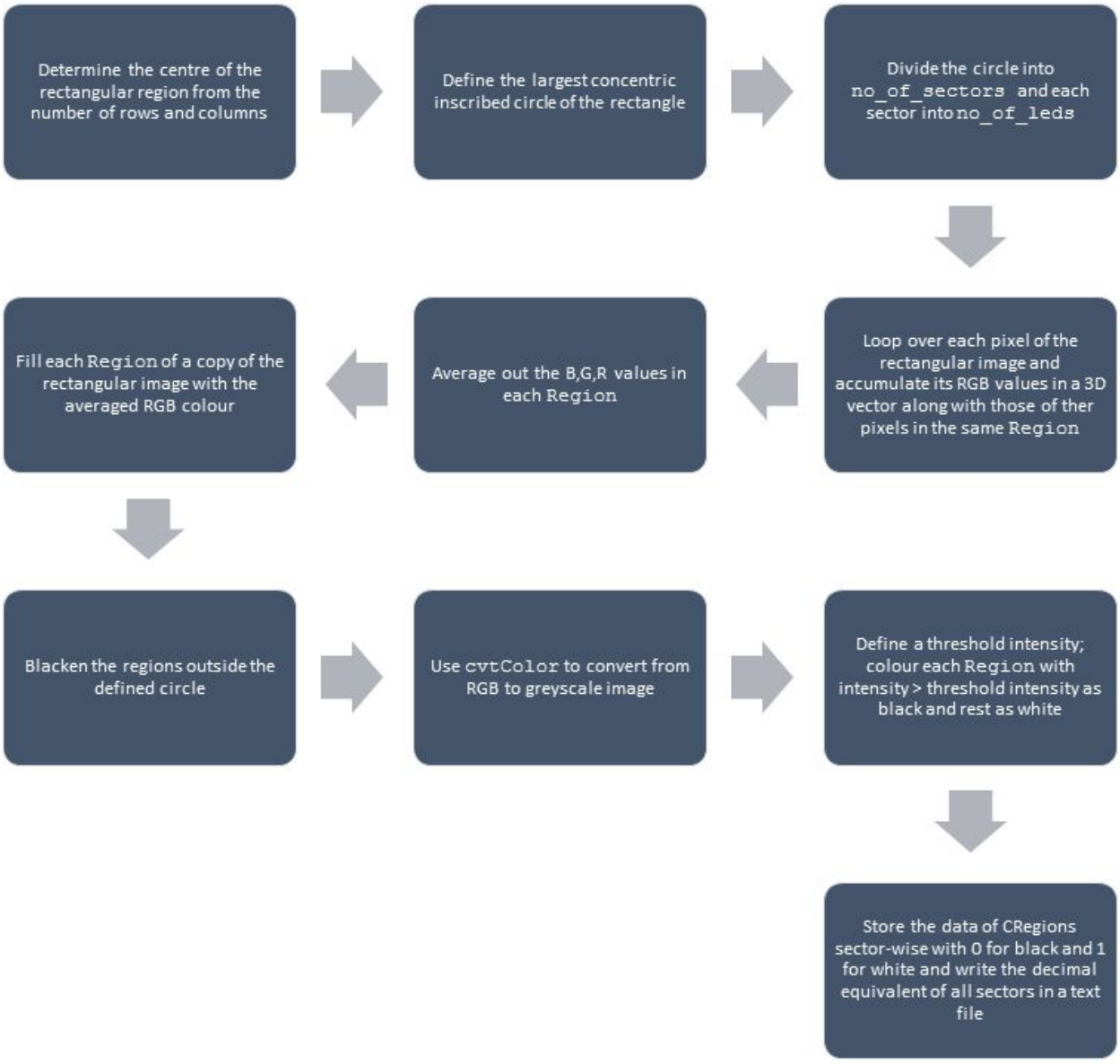


Image processing algorithm for 'static image' mode



Challenges and solutions:

	Challenge	Final Solution
1.	Since image processing code had been written first, we were not sure about the number of LEDs we would be using in the final project; the code, however, required this number extensively	We used dynamic memory allocation (in the form of vectors) instead of arrays to cater to the changing number of LEDs; provisions were also made for updating the entire code by altering the no. of LEDs at only one place
2.	The output of the image processing code written in C++ was a preview of the resulting image, and a data file to be fed to the Arduino; transferring this data file dynamically was another challenge	Since Python has a simple-to-use yet effective bluetooth library, we decided to transfer the data using a python script; it would read data from the data-file and send it to Arduino via bluetooth
3.	Integration of programs written in two different languages - C++ and Python - posed another challenge; we didn't want the user to manually execute the programs one after another	We developed a bash script (for Ubuntu) which would sequentially run the programs in required order, for seamless experience and integrity - running this script only once was sufficient
4.	The baud rate of HC-05 used was 38400. If we used normal data transfer technique, time required for the same would have been quite long, as well as amount of SRAM consumed would be higher	Data compression was implemented; the state of 16 LEDs ('on' implying 1, and 'off' implying 0) was converted into a binary string - the decimal equivalent of this string was sent via bluetooth. This drastically reduced the amount of data to be transferred at the cost of very little Arduino memory
5.	The SRAM limitation of Arduino is 2kb, meaning that we couldn't store data in large two-dimensional arrays (the same reason forced us not to use RGB LEDs or video-display)	• Variables were reused extensively • Appropriate data-types were used - int was replaced by byte wherever possible • We cut down a bit on resolution (changed the no_of_sectors from 360 to 255) resulting in about 50% reduction in memory

- The same data compression technique (used during data transfer via bluetooth) was used to store data.

6. Data could be fed to Arduino via bluetooth only once; after the first trial, no more images could be displayed unless the sketch was uploaded once again

We added a 'refresh' function to the sketch uploaded on Arduino, which would be triggered if 'r' is received via bluetooth; it brings back the Arduino to the original state, thus allowing the process to be repeated endlessly

Practical constraints

Video display

Though initially we planned to make video display as well, memory limitations of Arduino and the transfer rate of Bluetooth module limited our ability to display videos. 2kb of SRAM allowed the data of only one image to be stored at any instant. Moreover, sending data of different frames of a video via bluetooth to Arduino would make the video-display rather discontinuous (since data transfer took around 10 seconds), thereby spoiling the 'feel' of a video.

RGB display

Also, we had the programs ready for RGB display but due to some problems with shift resistor and memory shortage in the Arduino, we could not accomplish it.

Reference

Ohmbrew (inspiration for the project) : http://www.ohmbrew.net/prop/prop.shtml
OpenCV tutorial : http://docs.opencv.org/trunk/d9/df8/tutorial_root.html
Arduino tutorial : https://www.arduino.cc/en/Tutorial/BuiltInExamples
Stackoverflow links for PyBluez, bash scripting and other programming issues
74HC595 chip : https://www.arduino.cc/en/Tutorial/ShiftOut