JODHPUR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous College, Affiliated to Bikaner Technical University)

NAAC 'A' Grade | NBA Accredited | Ranked 1st in State

PRESENTATION OF MINI-PROJECT

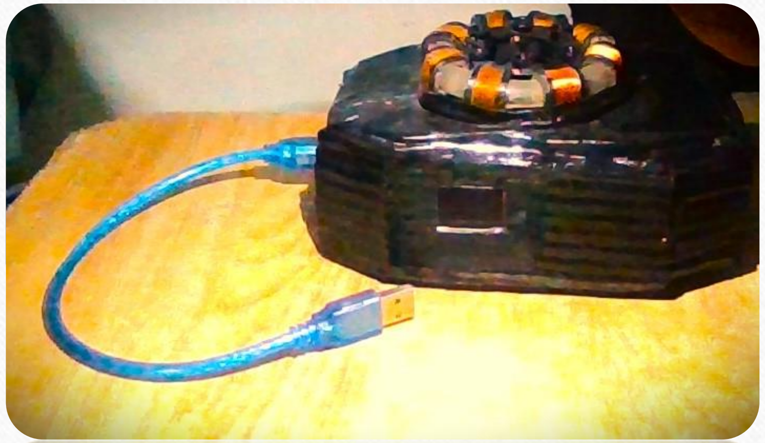
TITLE : ARC REACTOR CPU PERFORMANCE MONITOR WITH ARDUINO

PRESENTED BY:

NITESH SOLANKI	: T22EJIAI070
PRERNA ACHARIYA	:
PRINCE KUMAR	:T22EJAI080
SARVAJEET	:

GUIDE NAME

PROF. SUMIT



WHAT IS AN " ARC REACTOR CPU PERFORMANCE MONITOR"?

His handy desktop device runs a Python script on the monitored PC, which passes along CPU information over serial to an Arduino Uno. The Uno pulses the Arc Reactor in proportion to the computer load using a transistor, with higher frequency pulses indicating a heavily loaded CPU and lower frequencies for lower CPU usage. An OLED display is also implemented for numerical feedback, and everything is housed in a nice space house like structure.

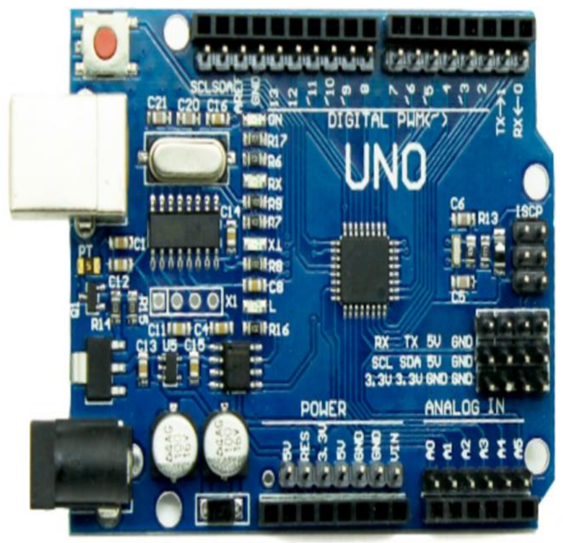
Components of Arc Reactor CPU Performance Monitor with Arduino.

Components	Quantity
#ARDUINO AND USB A TO B CABLE	1
#ARC REACTOR	1
#8050 NPN TRANSISTOR	1
#RESISTER(500OHM)	1
#OLED DISPLAY	1
#MALE AND FEMAL HEADER STRIPS	10,10
JUMPER WIRES	20

COMPONENT WORKING

ARDUINO UNO..

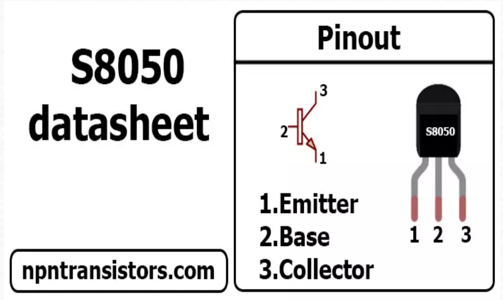
-THE ARDUINO IS AN OPEN SOURCE MICROCONTROLER BOARD BASED ON MICROCHIP ATMEGA 328P MICROCONTROLER AND DEVELOPED BY ARDUINO. THE BOARD IS EQUIPED SETS OF DIGITAL AND ANALOG INPUT/OUTPUT PINS THAT MAY BE INTERFACED TO VARIOUS EXPANSION BOARD AND OTHER CIRCUIT .



ARC REACTOR

THE ARC REACTOR WAS A FUSION TYPE POWER SOURCE FEATURING A PALLADIUM CORE,AND WAS THE INITIAL POWER SOURCE OF THE IRON MAN SUITS,IN THESE PROJECTS THE WORK OF ARC REACTOR IS TO GLOW LIGHT BRIGHT WHEN CPU LOAD IS INCRESE BY READ THE DATA FROM ARDUINO AND DEEM LIGHT OR DECREASE BRIGHTNESS WHEN CPU LOAD IS DECREASE.

NPN TRANSISTOR



S8050 is mainly used as Class B Push-Pull amplifiers. Class B push-pull amplifiers make use of two complementary transistors (one NPN + one PNP) connected in a fashion with the voltage source and load, as shown below. Both receive the same power signal but in opposite phases to each other.

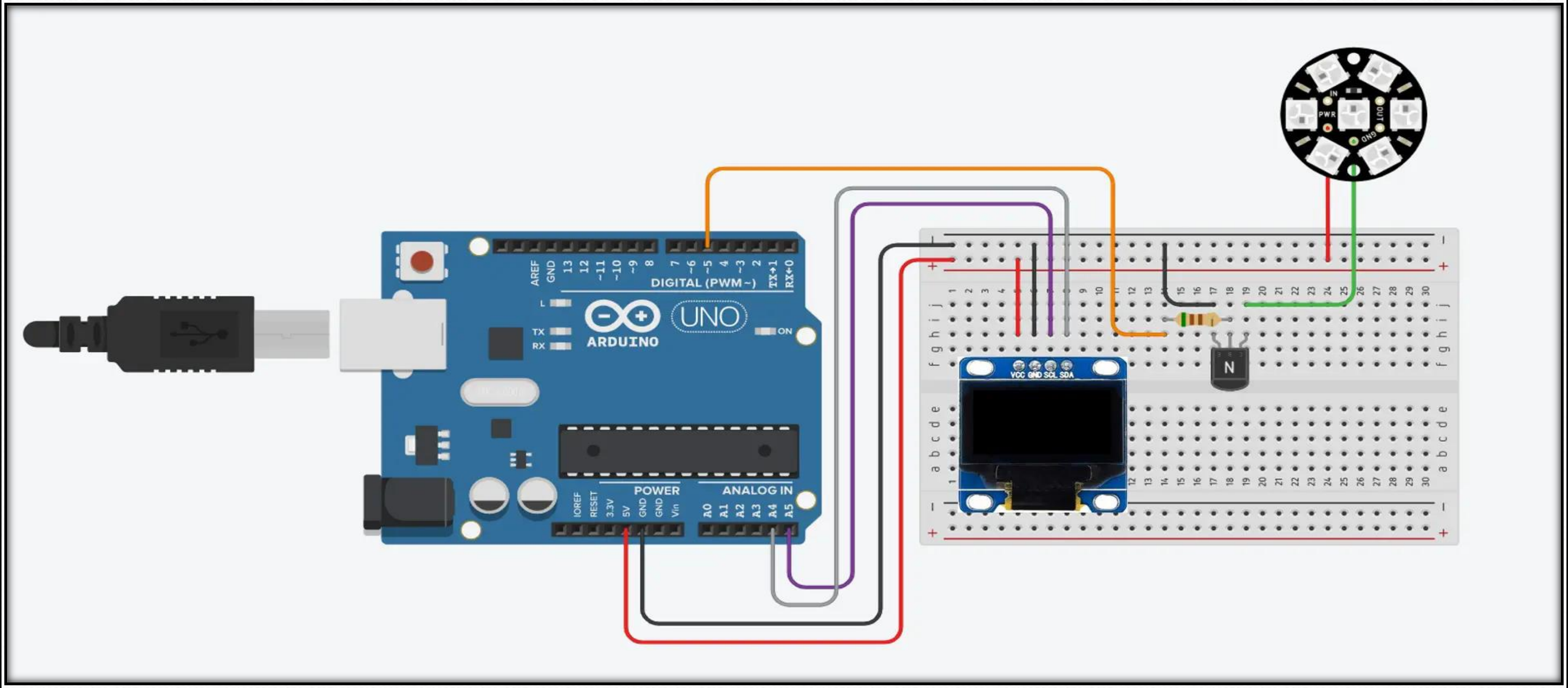
OLED DISPLAY

OLED TV is a television display technology based on the characteristics of **organic light-emitting diodes** (OLED). An OLED diode is made of six different layers with two of them retaining organic properties. When current is passed through these diodes, these organic layers produce light which passes through a colour refiner that produce picture on the screen. As these panels do not require a backlit setup, TV makers can eliminate the extra bulk and make OLED TVs significantly thinner than LED TVs. In fact, these panels are so compact that they are even used in premium phones. Now you have the same, brilliant performance in a TV.

JUMPERS

A jumper wire is an electrical wire or group of cables with a connector or pin at each end which is normally used to interconnect the component of a bread board or the other prototype or test circuit internally or with other Equipment or component without soldering.

LAYOUT DIAGRAM



PROGRAMMING THE ARC REACTOR CPU PERFORMANCE MONITOR

There are three things you need to set up to get your CPU data to be sent to your Arduino, a source of the data on your computer, a script to read the source on your computer and send the data through a serial communication port to your Arduino and the code on the Arduino to read the data, recompile it and then update the display and reactor LED.

1.>Start with the Arduino code

Programming The Arduino Uno

The Arduino’s code is fairly simple thanks to the OLED displays libraries. Most of the code is actually dedicated to turning the data received through the serial communication port back into usable data.

-We start off by importing the libraries necessary to control the OLED display, then set up the display parameters and create a display object.
-We then define the Arc reactor LED control pin and set parameters for the initial brightness and the fade amount or difference in light intensity for each cycle. I created the initial brightness variable as a non-zero value so that the LED doesn't turn off entirely between cycles, I found this "flashing" to be too distracting on my desk.
-We then create a character array to receive the serial communication data and create a variable to store the CPU load.
-In the setup function, we start the Serial communication, then establish a connection with the display and update it to display the initial CPU load, which will be zero. If the Arduino is not able to communicate with the display, the code will hang here and won't continue into the loop. This is good for troubleshooting and makes sure that your Arduino is able to communicate with the OLED display
-In the loop function, we start by updating the arc reactors brightness and then increment the brightness variable by the fade amount for the next loop cycle. We then have a check to see if the brightness has reached a maximum or minimum and if so, reverse the fade direction.
-We then check if any Serial communication data is available. If there is data available then it is read into the character array until the % sign is received, indicating that the data is complete.
-Data is transferred to the Arduino letter by letter in ASCII format, not as an integer. So each digit is received individually and the Arduino needs to know when the number is complete and then convert it into an integer. The rest of the code in this section does just that, converting each digit received into either hundreds, tens or units and then adding them together to form a complete integer which the Arduino can then quantify.
-We then update the display to reflect the new CPU load and then map the CPU load to a delay duration between cycles, which corresponds to a faster or slower arc reactor pulse. The higher the CPU load, the faster the arc reactor will pulse.
-Lastly, we have a function to update the display which just clears the previous display and displays the new CPU load.
-While you've got your Arduino IDE open, also make a note of which com port is assigned to your Arduino Uno, as you'll need to update this in the python script.

2.>GETTING YOUR CPU LOAD

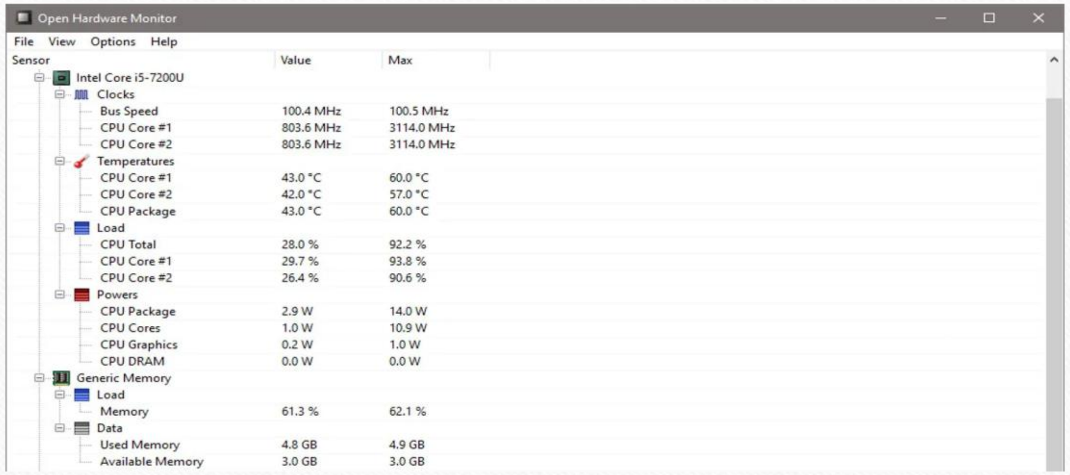
-In order to send the CPU load to your Arduino, we first need to be able to access it. One of the easiest ways to do this is using a freeware application called Open Hardware Monitor

-The Open Hardware Monitor app runs in the background on your PC and minimises to the system tray. It collects information on a number of different hardware items on your computer and makes it available to be read using a Python script.

-You'll need to make a note of your processor name displayed in the monitor as you'll need to update this in the python script in order to access your CPU load.

You'll also need to set the Open Hardware Monitor to run on startup so that you don't need to open it every time you restart your PC. To do this, go to options and make sure that "Start Minimized", "Minimize To Tray" and "Run on Windows Startup" are all checked. Then minimise the app.

how the open hardware monitor look likes..



Sensor	Value	Max
Intel Core i5-7200U		
Clocks		
Bus Speed	100.4 MHz	100.5 MHz
CPU Core #1	803.6 MHz	3114.0 MHz
CPU Core #2	803.6 MHz	3114.0 MHz
Temperatures		
CPU Core #1	43.0 °C	60.0 °C
CPU Core #2	42.0 °C	57.0 °C
CPU Package	43.0 °C	60.0 °C
Load		
CPU Total	28.0 %	92.2 %
CPU Core #1	29.7 %	93.8 %
CPU Core #2	26.4 %	90.6 %
Powers		
CPU Package	2.9 W	14.0 W
CPU Cores	1.0 W	10.9 W
CPU Graphics	0.2 W	1.0 W
CPU DRAM	0.0 W	0.0 W
Generic Memory		
Load		
Memory	61.3 %	62.1 %
Data		
Used Memory	4.8 GB	4.9 GB
Available Memory	3.0 GB	3.0 GB

3.>POSTING THE DATA TO YOUR ARDUINO USING A PYTHON SCRIPT

-Lastly, we have a python script which transmits the CPU load through the serial com port to your Arduino. The script reads the data posted by the Hardware monitor app, finds the CPU load and then posts the value to the Arduino.
-I didn't write this code. I've modified the code from Leonidas Tsekouras's Arduino PC Monitor example in which he sends a range of data to be displayed on an external LCD display.
-You'll need to update the IP address that your Open Hardware Monitor is posting the data too. This can be found in the Open Hardware Monitor app by going to Options -> Remote Web Server -> Port.
-You'll need to update the IP address that your Open Hardware Monitor is posting the data too. This can be found in the Open Hardware Monitor app by going to Options -> Remote Web Server -> Port.
-Both the python script and then hardware monitor can be set to automatically startup with your computer and run in the background, so you don't have to keep running them every time you turn your computer off. The python script can be automatically started using the windows "Start a Program" utility.

OUR FINAL STEP:-USING THE ARC REACTOR CPU MONITOR

-Upload the code your Arduino and make sure that the Open Hardware Monitor app is running on your computer and then run the Python script to start posting the data to your Arduino. You should then start seeing updates to your CPU performance on the OLED display and the reactor pulsing accordingly.

-On start-up, with no data being transmitted, the OLED display should still display CPU Load 0% and the reactor should be pulsing slowly.

-If you open up a benchmarking application to increase the CPU load to 100%, you will see it pulse really quickly.

-You can make your own adjustments to the pulse durations and brightness levels in the code. I found it too distracting to have the pulse turn off completely each cycle, so I limited the minimum brightness and also set the low CPU usage pulse duration to be quite long. You can edit these by making changes to the mapped delay, to the brightness variable and to the fadeAmount variable.

The end..

THANKYOU