


GSoC'24 Project Proposal

Sugar Labs: Sugar On Raspberry Pi

Anurag Singh



1. Basic Details

Name:

Anurag Singh

Contacts:

Email: principialquantum30@gmail.com

GitHub Profile: [Ovalelephant35](#)

Matrix Username: Anurag Singh (Ovalelephant)

LinkedIn Profile: [Anurag Singh](#)

LeetCode : [Ovalelephant35](#)

Your First Language:

I have proficiency in English and elementary proficiency in French, although Hindi is my first language.

Location and Time zone:

Location: Jaipur, Rajasthan, India

Time zone: Indian Standard Time (UTC+5:30)

Communication:

- UTC 02:00 – UTC 07:00
- UTC 08:00 – UTC 15:00
- UTC 16:00 – UTC 19:30

I am quite Flexible with any time if it helps in better communication with developers and mentors, and I will be reachable anytime through my Mobile and Email.

Education Details:

Currently, I am in my third year of academic pursuit, enrolled in a double major program encompassing Computer Science and Physics at BITS Pilani.

My introduction to programming occurred during my second year of study. Since then, I have delved into diverse domains including Data Structures, Computer Architecture,

System Design, Digital Electronics, and Object-Oriented Programming. Engaging in practical projects and coursework has afforded me a robust comprehension of these concepts, concurrently enhancing my aptitude for problem-solving.

2. Share Links, of your previous work on opensource projects.

For the past 7-8 months, I have actively engaged in contributing to open-source projects. Along this journey, I have acquired valuable knowledge spanning embedded systems, system design, object-oriented programming, machine learning, Arduino, and Raspberry Pi.

Additionally, I have tackled numerous algorithmic challenges on platforms such as Codeforces and LeetCode, nurturing a keen aptitude for programming and problem-solving in a broader context.

Projects/Repository/Pull Requests Links	Description
100-RPi-Projects	Compilation of Raspberry Pi projects.
Verilog-Digital-Design	Designing Systems using Verilog.
Computational-Physics	Utilizing Python for Computational Physics.
Disco-Graph-Optimization-Project	Course Assignment – Bipartite Tough
JavaScript-Projects	Projects implemented in JavaScript.
ESG	ESG – Capability Build
FreeCad	FreeCad – API Integration
Chatbots	Chatbot with Self-Learning Capabilities.
Competitive-Programming-Solution	Challenges in Algorithmic Problem Solving.

I have made over **1050 contributions** on GitHub, including more than **40 pull requests** and involvement in over **30 issues**, both active and resolved. My contributions have extended across various organizations and projects.

For further details, please visit my profile at [Ovalelephant35](#) on GitHub.

3. Convince us that you will be a good fit for this project, by sharing links to your contribution to Sugar Labs

During the past four months, I've been an active member of the Sugar community, immersing myself in its ethos and methodologies. Throughout this period, I've gained substantial insights into the workings of Sugar activities, actively contributing to various activities and diligently addressing numerous issues and bugs. This journey has been a profound learning experience, enriching my understanding of Sugar's core functionalities and fostering my growth within the community.

Here are the pull requests I have generated for Sugar Labs:

Pull Request Link	Description	Status
<u>#2</u>	Added an installation instruction file for Raspberry Pi.	Merged
<u>#3</u>	Added a Virtual Machine guide and a Hardware Guide specifically tailored for Raspberry Pi.	Merged
<u>#4</u>	Updated the documentation to include information about auto-logging and the Desktop Image specifically for Raspberry Pi.	Merged
<u>#5</u>	Added Rpi-soas.md, a guide specifically for Fedora Sugar on a Stick (SoaS) on Raspberry Pi.	Merged
<u>#982</u>	Modified the Raspberry Pi documentation in Sugar to include RPi-docs.	Merged
<u>#36</u>	Added labels to the pie chart for better clarity in the finance activity.	Merged
<u>#3</u>	Ported the PyEyes activity to Python 3.	Merged
<u>#20</u>	Resolved the issue that was hindering the opening of the Cardsort activity.	Merged
<u>#15</u>	Ported the Panorama activity to Python 3.	Open
<u>#12</u>	Improved NumberRush Activity Gaming Experience by adding bonus points.	Open
<u>#19</u>	Added a counter to Flip Activity for better user experience.	Open

#8	Added Names to Planets and Sun in Solar-System Activity.	Open
#24	Updated Name of Countries in Countries-Activity	Open

Made over **40 commits** and thoroughly documented all possible combinations of devices and environments in which Raspberry Pi and Sugar can be integrated. Additionally, I've tested over **80 activities on Raspberry Pi** and addressed more than **20 issues.**

Here are the Issues I have Worked on in Sugar Labs:

Issue Link	Description	Status
#131	Fix Sugar Browser activity failure on RPi.	Closed
#13	Panorama activity ported to Python3	Closed
#2	PyEyes activity ported to Python3	Closed
#19	Fix CardSort activity failure on RPi.	Closed
#126	Could not determine the accessibility bus address.	Open
#19	Measure activity failing to Run on RPi	Open
#20	Score Empty in Flip Activity.	Open
#3	Port Micropolis activity to Python 3.	Open
#10	Unable to Run MathGraph32 activity.	Open
#10	Unable to Run Paths activity on RPi.	Open
#15	Unable to Run I-can-Read-activity on RPi.	Open
#27	Abacus activity Margin Improvement	Open
#21	Internationalization in Country Activities	Open

In addition to these tasks, I've collaborated with several other organizations, including **QEMU, VirtualBox, VMware, and the Raspberry Pi/Raspberry Pi Stack Exchange** communities, to address various inquiries and related matters.

I've also engaged with matters related to Sugar Labs, assisting with running it in different environments and addressing related issues.

4. Project Details

How I got Interested in this Project:-

While browsing the internet for open-source organizations, I stumbled upon a [Wired](#) article that caught my attention. Further research on platforms like [Twitter](#) led me to discover Sugar Labs, which immediately piqued my interest. Upon exploring the previous year's GSoC idea list, I found the concept of integrating Sugar on Raspberry Pi, which seemed like the perfect opportunity to combine the best of both worlds. I embarked on my journey by familiarizing myself with the Sugar environment, understanding its features, and setting up the development environment. Subsequently, I undertook the task of implementing Sugar on Raspberry Pi across a variety of devices and virtual machines, including different operating systems such as Windows, macOS, and Linux, as well as virtualization platforms like QEMU, VirtualBox, and VMWare.

What are You Making?

This project comprises **three** crucial components:

1. Understanding **Raspberry Pi** and **Components**:

- Explore Raspberry Pi hardware and peripherals like GPIO, sensors, and actuators for potential utilization.

2. Exploring **Sugar Activities** Integration:

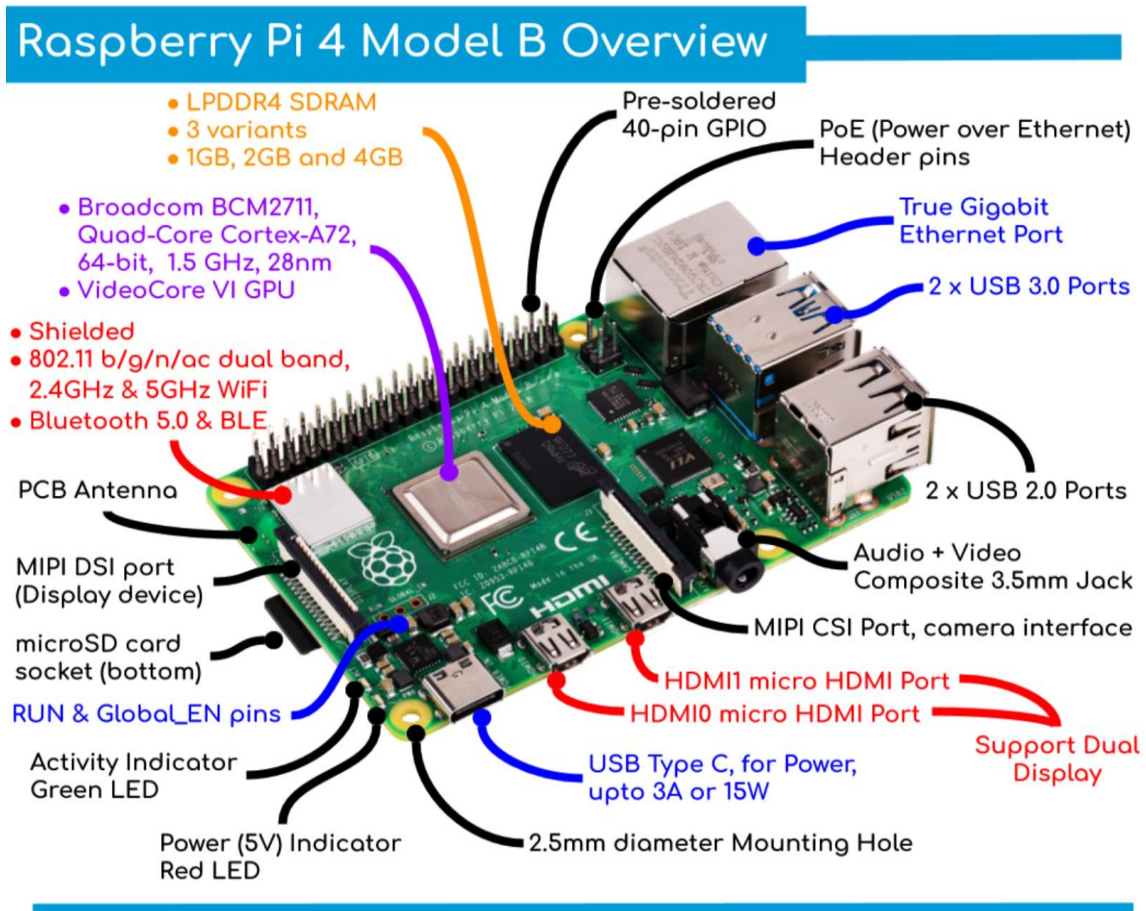
- Learn how to integrate Sugar Activities seamlessly into the Raspberry Pi environment and fine tuning them for optimized performance.

3. **Packaging Sugar** on Raspberry Pi:

- Package Sugar for Raspberry Pi akin to Fedora SOAS and create detailed **RPI-documentation** for easy deployment.

Part-1 Understanding RPi, Its Component and Different Peripherals: -

We'll commence with an overview of the Raspberry Pi 4, our primary component alongside the RPi Pico, for our project.



Here is the detailed description for Raspberry Pi Components:-

Key Components	Description
Processor	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
RAM	1GB, 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM (depending on model)
Wireless	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
Networking	Gigabit Ethernet
USB Ports	2 USB 3.0 ports; 2 USB 2.0 ports
GPIO Header	Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
Video Output	2 × micro-HDMI ports (up to 4kp60 supported), 2-lane MIPI DSI display port

Camera Port	2-lane MIPI CSI camera port
Audio/Video Ports	4-pole stereo audio and composite video port
Video Codecs	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
Graphics APIs	OpenGL ES 3.1, Vulkan 1.0
Storage	Micro-SD card slot for loading operating system and data storage
Power Supply	5V DC via USB-C connector (minimum 3A*), 5V DC via GPIO header (minimum 3A*), Power over Ethernet (PoE) enabled (requires separate PoE HAT)
Operating Temperature	0 – 50 degrees C ambient

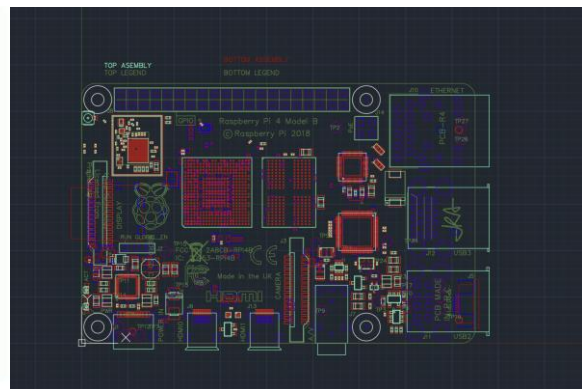
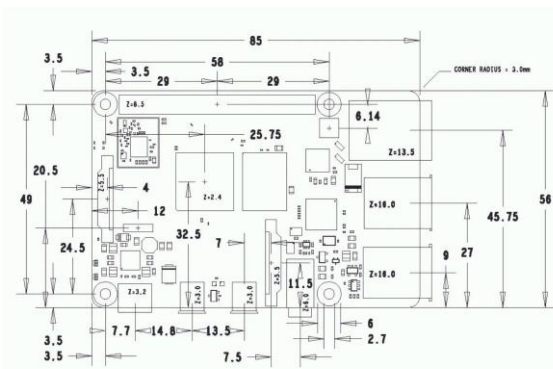
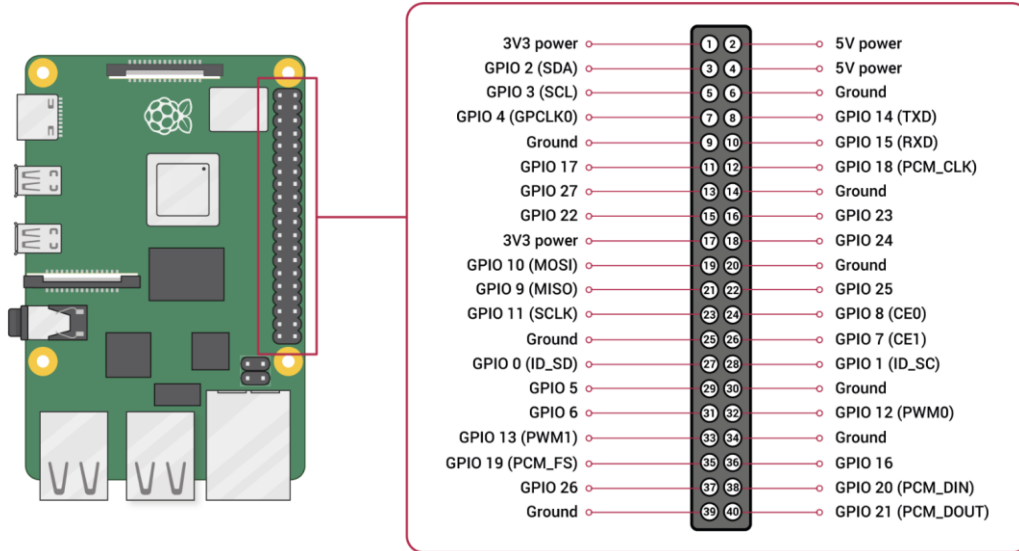


Fig 1.0 is mechanical layout and Fig 2.0 is engineers drawing representation made using Freecad.


GPIO (General Purpose Input/Output) Pins:



GPIO pins on the Raspberry Pi allow it to interact with external components such as sensors, LEDs, buttons, and more. The GPIO header consists of 40 pins.



Here is Detailed Overview of all the peripherals and components:-

Key Peripherals	Description
<p data-bbox="264 1193 724 1227">Temperature/Humidity Sensor</p> 	<p data-bbox="826 1308 1369 1413">Measures temperature and humidity levels in the surrounding environment.</p>
<p data-bbox="344 1581 644 1615">Motion Sensor (PIR)</p>	<p data-bbox="826 1767 1374 1832">Detects motion by sensing changes in infrared radiation.</p>



Light Sensor (LDR)

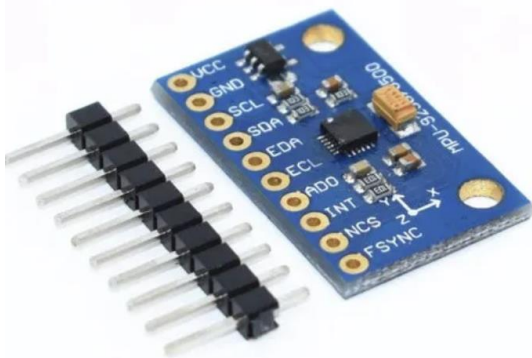


Ultrasonic Distance sensor



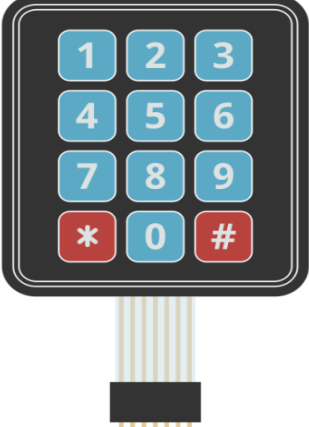


Measures distance by emitting ultrasonic waves and calculating the time it takes for the waves to bounce back.

Accelerometer



Measures acceleration and angular velocity, used for orientation sensing.

Number Pad / Keypad

	<p>A matrix of buttons used for inputting numerical or alphanumeric data. Each button press corresponds to a unique GPIO pin.</p>
<p>Joystick</p> 	<p>A control input device consisting of a stick that pivots on a base and typically has buttons. Joystick movements and button presses can be translated into GPIO signals.</p>
<p>LED Matrix Display</p> 	<p>A grid of LEDs that can display text, numbers, and simple graphics.</p>

All these components and peripherals are either included in a set with the RPi or can be purchased separately for just a dollar or two more.

Part -2 Exploring Sugar Activities and its integration:-

I've tested over **80 activities**, with some already finalized. However, as I continue testing, the comprehensive list provided here may undergo changes in the coming days.

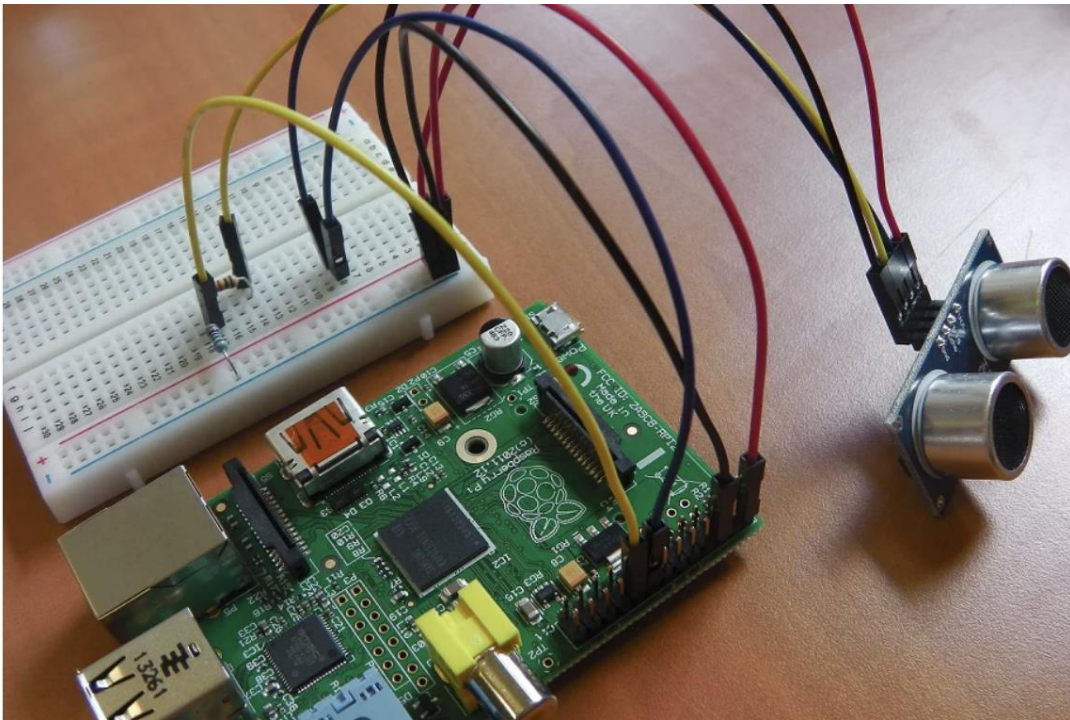
1. TurtleBlocks Activity: -

Key documents serving as sources of inspiration for ideas include: -

1. https://wiki.sugarlabs.org/go/Activities/Turtle_Art/Plugins - Author Walter Bender
2. <https://wiki.sugarlabs.org/go/User:Walter/TurtleArt> - Author Walter Bender
3. <https://github.com/walterbender> - Author Walter Bender
4. <https://activities.sugarlabs.org/en-US/sugar/addon/4027> - Sugar labs App
5. https://wiki.sugarlabs.org/go/Activities/Turtle_Art/Packaging - Author Walter bender

Sensors, Peripherals to be utilised	Description and Integration
1. Ultrasonic Distance Sensor	<p>Plugin/Extension: Develop a plugin or extension for Turtle Blocks that provides blocks to interact with the Ultrasonic Distance Sensor. These blocks can include functionalities such as reading distance values and triggering actions based on proximity.</p> <p>External Library: Utilize a Python library compatible with the Ultrasonic Distance Sensor, such as RPi.GPIO, to interface with the sensor. Users can write Python code within Turtle Blocks to access sensor data and control project behaviour accordingly.</p>
2. Camera Apparatus Peripheral	<p>Plugin/Extension: Develop a plugin or extension that incorporates blocks for accessing the camera module. These blocks can include commands to capture images/videos, display live feed, and manipulate captured media.</p> <p>External Library: Utilize Python libraries like pi camera to interact with the camera module. Users can write Python scripts within Turtle Blocks or import functions from external modules to handle camera functionalities.</p>
3. Motion Detector (PIR) Sensor	<p>Plugin/Extension: Create blocks within Turtle Blocks that interface with the Motion Detector (PIR). These blocks can include commands to monitor motion</p>

status, set detection parameters, and execute actions upon detection.
External Library: Utilize GPIO libraries like RPi.GPIO in Python to interface with the Motion Detector (PIR). Users can write Python scripts to continuously monitor the sensor status and respond to motion events within their Turtle Blocks projects.



Setting Up distance detector for TurtleBlocks Activity

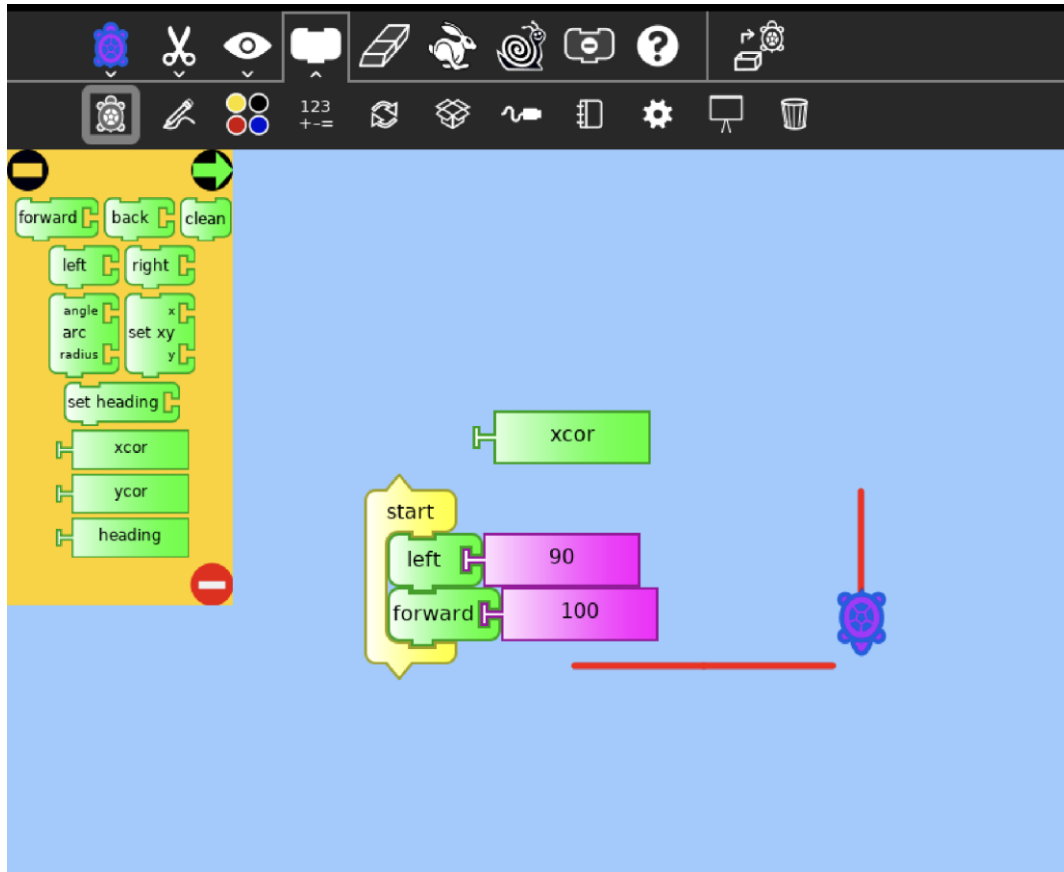
```
import RPi.GPIO as GPIO
import time

# Set up Ultrasonic Distance Sensor GPIO pins
TRIG = 23
ECHO = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def measure_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    start_time = time.time()
    while GPIO.input(ECHO) == 0:
        start_time = time.time()
    while GPIO.input(ECHO) == 1:
        stop_time = time.time()
    elapsed_time = stop_time - start_time
    distance = (elapsed_time * 34300) / 2 # Speed of sound is 343 m/s
    return distance

# Example usage:
distance = measure_distance()
print("Distance:", distance, "cm")
```

Sample Code Snippet to take input from Sensor



TurtleBlocks Activity on RPi

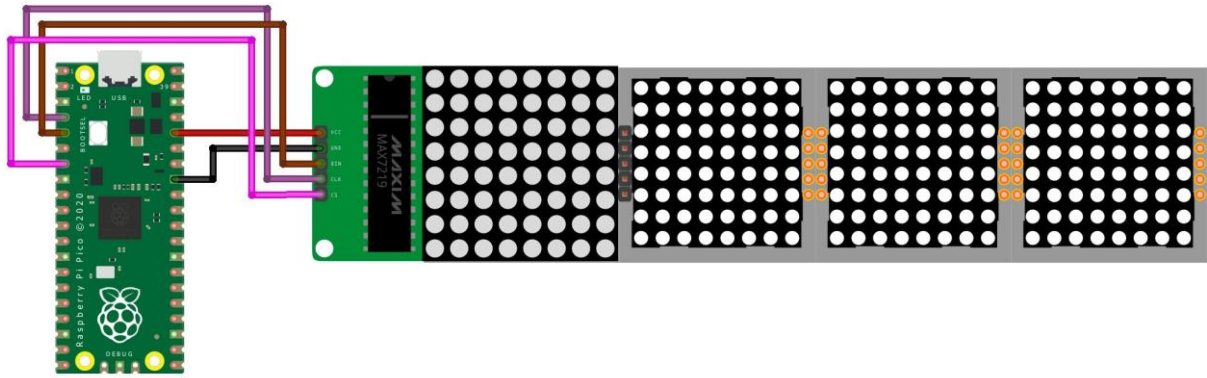
2. Measure Activity:-

Key Peripherals	Description/Integration
<p>1. Microphone</p>	<p><u>Plugin/Extension Development:</u> Develop a plugin or extension for the Measure activity that interfaces with the microphone. Include functionalities for capturing audio signals from the microphone and displaying waveforms or frequency spectra within the Measure activity interface.</p> <p><u>Python Scripting:</u> Utilize Python scripts to control the microphone. Within the Measure activity, write scripts to capture audio signals from the microphone and visualize them in real-time.</p>
<p>2. LED Matrix Display</p>	<p><u>Plugin/Extension Development:</u> Develop a plugin or extension for the Measure activity that interfaces with the LED matrix display.</p>

	<p>Include functionalities for displaying waveforms, frequency spectra, and statistical representations of signals on the LED matrix.</p> <p><u>Python Scripting:</u> Utilize Python scripts to control the LED matrix display. Within the Measure activity, write scripts to translate captured signals into visual representations on the LED matrix display.</p> <p><u>External Libraries:</u> Use Python libraries compatible with LED matrix displays, such as RPi.GPIO and libraries specific to the LED matrix hardware. Import functions from these libraries within the Measure activity to control the LED matrix display and render visualizations of signals and data.</p>
--	--



Measure Activity



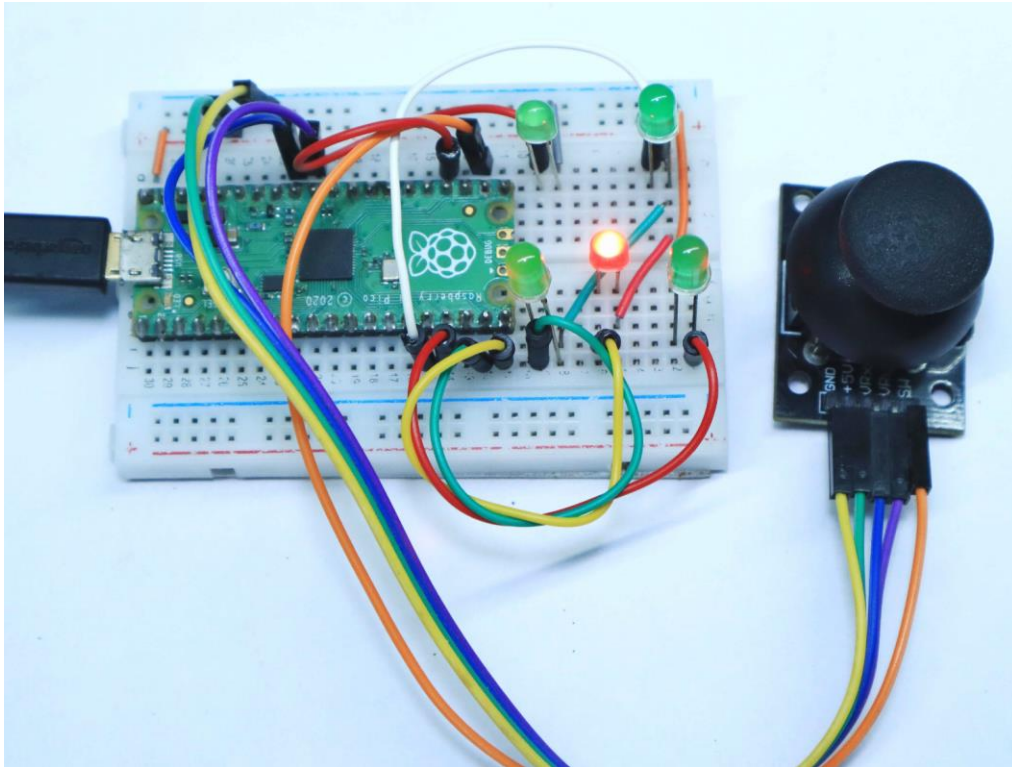
Matrix display for Visualization

3. Physics Activity.

A pre-existing plugin integrated into the Physics activity allows for the integration of a joystick to control motion, enhancing the user experience with real-time interaction.

Key Peripheral	Description
<p style="text-align: center;">Joystick</p>	<p><u>Hardware Connection:</u> Connect the joystick to the microcontroller or platform. Typically, a joystick has analog outputs for its axes (X and Y) and digital outputs for buttons.</p> <p><u>Analog-to-Digital Conversion (ADC):</u> Read the analog signals from the joystick's X and Y axes using the microcontroller's ADC peripheral. Convert the analog voltages to digital values representing the joystick position.</p> <p><u>Digital Input/Output (GPIO):</u> Read the digital signals from the joystick buttons using the microcontroller's GPIO pins. Each button press or release should be detected and processed in the assembly code.</p>

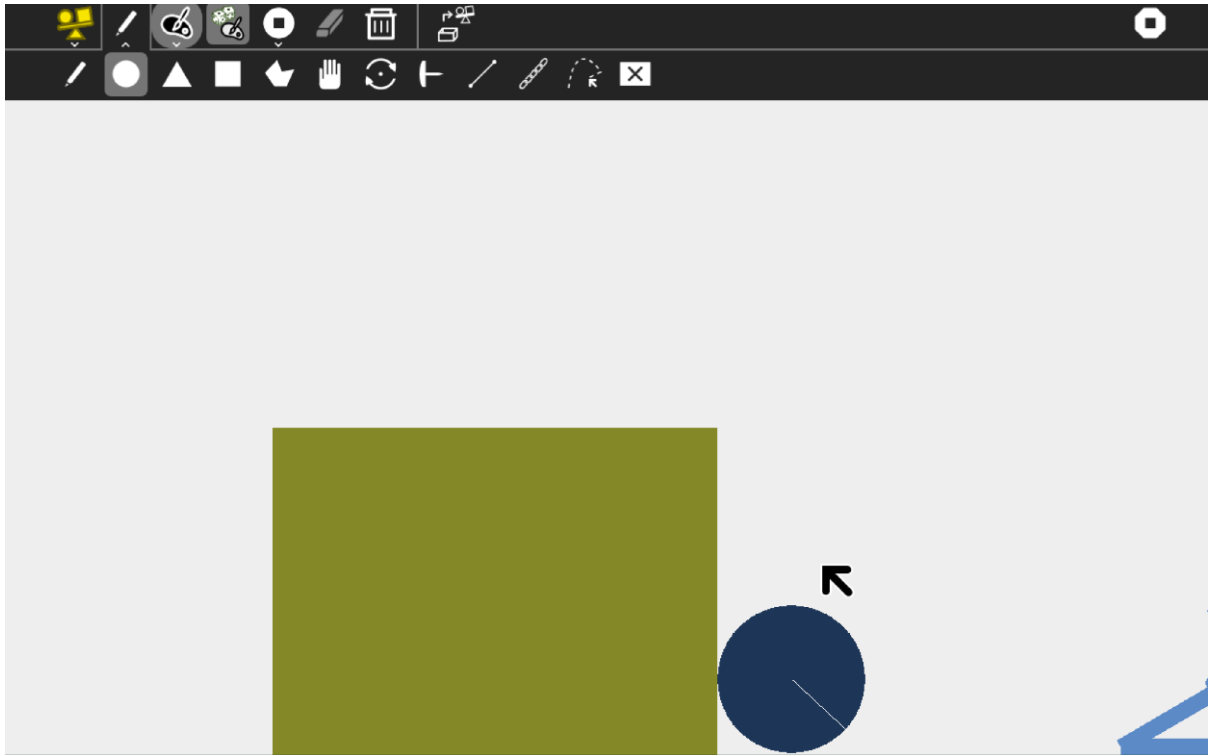
To make a proper plugin/extension with proper instruction to integrate joystick as input



RPi Pico Setup for Joystick

```
0x09, 0x04, // Usage (Joystick)
0xA1, 0x01, // Collection (Application)
0x15, 0x81, // Logical Minimum (-127)
0x25, 0x7F, // Logical Maximum (127)
0x09, 0x01, // Usage (Pointer)
0xA1, 0x00, // Collection (Physical)
0x09, 0x30, // Usage (X)
0x09, 0x31, // Usage (Y)
0x75, 0x08, // Report Size (8)
0x95, 0x02, // Report Count (2)
0x81, 0x02, // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)
0xC0, // End Collection
0x05, 0x09, // Usage Page (Button)
0x19, 0x01, // Usage Minimum (0x01)
0x29, 0x08, // Usage Maximum (0x08)
0x15, 0x00, // Logical Minimum (0)
0x25, 0x01, // Logical Maximum (1)
0x75, 0x01, // Report Size (1)
0x95, 0x08, // Report Count (8)
0x81, 0x02, // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)
0xC0, // End Collection
```

Assembly Code Input to Control Joystick in 2-D Axis



Physics Activity on RPi using Joystick

4. Pippy Activity

Pippy stands out as one of the most significant activities that can be utilized effectively.

Key Usage	Description
Python Programming Practice	Pippy allows users to write and execute Python code directly on the Raspberry Pi. This provides an excellent platform for practicing Python programming skills, learning new concepts, and experimenting with code.
Project Development	It provides a platform for users to develop Python-based projects on the Raspberry Pi. Users can create applications, games, utilities, and more using Python programming language and various libraries available on Raspberry Pi OS.
Integration with Raspberry Pi GPIO	Since Pippy runs on Raspberry Pi OS, it can easily integrate with the GPIO pins of the Raspberry Pi. Users can write Python code in Pippy to interact with sensors, LEDs, motors, and other hardware

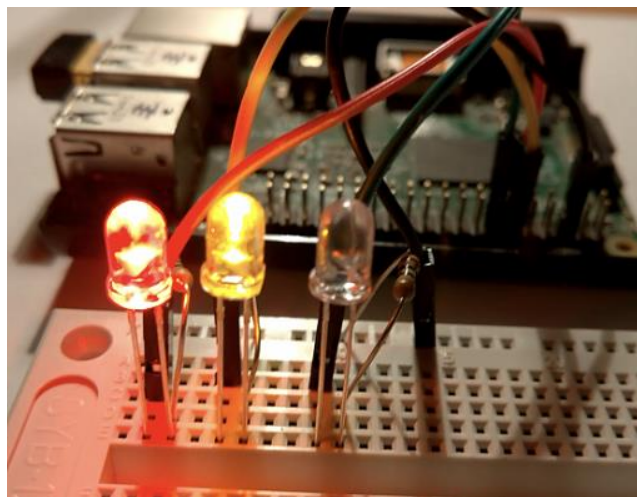
	components connected to the GPIO pins, enabling hands-on learning of physical computing concepts.
Collaborative Learning	Pippy supports collaborative learning and sharing of projects within the Sugar learning platform. Users can share their Python code, projects, and experiences with peers and educators, fostering collaboration and knowledge exchange.

```

1 import time
2 Green = 0
3 Yellow = 1
4 Red = 2
5
6 Green_duration = 5
7 Yellow_duration = 2
8 Red_duration = 5
9
10 def display_traffic_light(state)
11     if state==Green:
12         print("GREEN")
13     if state==Yellow:
14         print("Yellow")
15     if state==Red:
16         print("Red")
17
18
19 def main():
20     while true:
21         display_traffic_light(Green)
22         time.sleep(Green_duration)
23
24         display_traffic_light(Yellow)
25         time.sleep(Yellow_duration)
26
27         display_traffic_light(Red)
28         time.sleep(Red_duration)
29

```

Sample Code to Control Traffic light signals implemented using pippy though we need extension/plugin to implement it directly as input rather currently it acts as text-editor

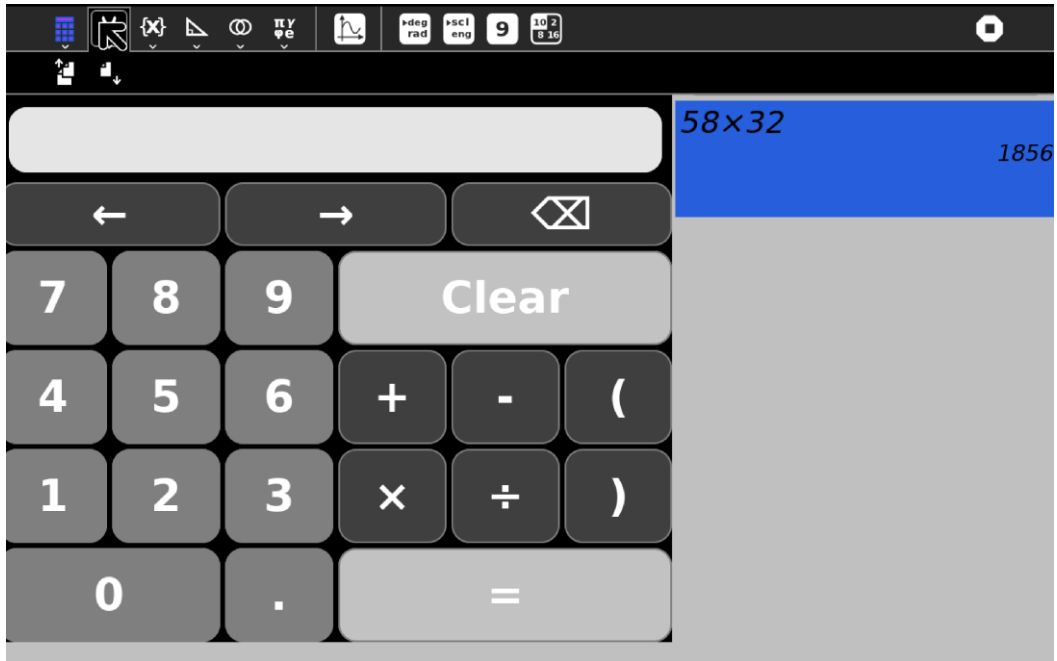


Traffic light implementation

5. Calculator

Integration	Description
Number Pad Connection	Develop a Python script that interacts with the Calculator activity. This script should receive input from the number pad and send appropriate commands to the Calculator activity to perform calculations.
GPIO Interface	If you're using a GPIO-based number pad, connect it to the GPIO pins of the Raspberry Pi. Ensure proper wiring and configuration to interface with the GPIO pins.





Calculator Activity on RPI

```
import RPi.GPIO as GPIO
import time

# GPIO pins for the number pad buttons
BUTTONS = {
    '1': 17, '2': 27, '3': 22,
    '4': 18, '5': 23, '6': 24,
    '7': 5, '8': 6, '9': 13,
    '0': 12, '+': 16, '-': 20,
    '*': 21, '/': 25, 'C': 26,
    '=': 19
}

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
for pin in BUTTONS.values():
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Calculator state
expression = ''
result = ''
```

Leveraging GPIO pins to input different number pad button

Below is the comprehensive list of activities that I will be setting up on Raspberry Pi to leverage sensors, fine-tune, and provide detailed instructions in RPI-docs:

Abacus Activity	I can read activity
Panorama Activity	Color deducto activity
Finance Activity	Connect the color activity
Countries Activity	Fraction Bounce activity
Solar-System Activity	Slide
Labyrinth activity	CardSort
2-Cars Activity	Number-rush activity
Pukklanapac	Apple and Haken
Ruler	Paths
Read	Write

This draft list of activities is subject to change as more activities are yet to be tested. There may be additions and subtractions to this list as testing progresses.

Below are the two proposed games that I suggest. If time allows, these games will be specially designed for Raspberry Pi users, providing an immersive learning experience and interactive gameplay:

1. Number Guessing Game using Number Pad:

Hardware Setup:

NumberPad Connection:

Connect the NumberPad to the Raspberry Pi via GPIO pins or USB interface.

Software Implementation:

Initialize GPIO Pins:

Set up GPIO pins to read input from the NumberPad buttons.

Generate Random Number:

Generate a random number within a predefined range (e.g., 1 to 100) as the target number for the player to guess.

Read NumberPad Input:

Continuously read input from the NumberPad to capture the player's guesses.

Compare Guess with Target:

Compare each guess made by the player with the target number.

Provide feedback to the player (e.g., "Too high", "Too low", "Correct!") based on the comparison.

End Game:

End the game when the player guesses the correct number or after a certain number of attempts.

Optionally, allow the player to restart the game after completion.

2. Temperature Conversion Game:

Software Implementation:

User Interface:

Design a user interface to display temperature values in Celsius, Fahrenheit, and Kelvin. Include input fields or buttons for the user to enter temperature values.

Read User Input:

Read temperature values entered by the user.

Convert Temperatures:

Implement algorithms to convert temperature values between Celsius, Fahrenheit, and Kelvin.

Provide options for the user to select the conversion direction (e.g., Celsius to Fahrenheit, Fahrenheit to Celsius, etc.).

Display Results:

Display the converted temperature values in the user interface.

Update the display dynamically as the user enters new temperature values or selects different conversion options.

Validation and Error Handling:

Validate user input to ensure it is within reasonable temperature ranges and follows the correct format.

Handle errors gracefully and provide feedback to the user in case of invalid input or calculation errors.

Interactive Learning:

Provide additional information or explanations about temperature units and conversion formulas to facilitate learning.

Incorporate interactive elements such as quizzes or challenges to engage user

Part - 3 Packaging Raspberry Pi and launching it alongside all the essential activities is a key step in making it accessible and user-friendly.

Document / Ideas Source	Description
RPi-Docs	This guide will cater to users interested in leveraging Sugar's interactive learning activities for educational purposes. The documentation will cover the setup process, from installing Raspberry Pi OS to configuring Sugar Desktop Environment . Additionally, it will provide detailed instructions on exploring Sugar activities, including

	<p>launching the environment, navigating through the available activities, and utilizing them effectively for interactive learning experiences. Furthermore, the documentation will include tips and best practices for integrating Sugar activities with Raspberry Pi projects, enhancing the educational potential of these versatile devices.</p>
<p>https://docs.fedoraproject.org/en-US/packaging-guidelines/ Fedora Spin (Guidelines for Packaging)</p>	<p>For the Raspberry Pi documentation project, I propose to draw inspiration from the Fedora Spin Sugar on a Stick (SoaS) as a reference for packaging practices, licensing considerations, and overall documentation structure. By examining the packaging process of Sugar on a Stick within the Fedora ecosystem, we can gain valuable insights into the organization, documentation standards, and licensing requirements pertinent to educational software distributions like Sugar. This will inform the documentation's approach to packaging Sugar for Raspberry Pi OS, ensuring adherence to licensing agreements, clear documentation of dependencies, and transparent distribution practices</p>
<p>https://github.com/sugarlabs/sugar/blob/master/docs/development-environment.md & https://github.com/sugarlabs/sugar/blob/master/docs/debian-packaging-example.md</p>	<p>These two documents detail the packaging process for Sugar activities and how they are natively packaged.</p>
<p>https://opensource.com/article/21/7/custom-raspberry-pi-image (Create Your Own Custom Image)</p>	<ul style="list-style-type: none"> • Stage 0: Bootstrap—Creates a usable filesystem • Stage 1: Minimal system—Creates an absolute minimal system • Stage 2: Lite system—Corresponds to Raspberry Pi OS Lite

	<ul style="list-style-type: none">• Stage 3: Desktop system— Installs X11, LXDE, web browsers, and so on• Stage 4: Corresponds to an ordinary Raspberry Pi OS• Stage 5: Corresponds to Raspberry Pi OS Full
--	--

There is a significant learning curve associated with packaging and deploying, and it will be meticulously addressed to ensure that the process serves as a comprehensive solution for individuals, facilitating their seamless entry into the realm of Sugar on Raspberry Pi.

How will it impact Sugar Labs?

Sugar on Raspberry Pi will significantly impact Sugar Labs by extending its educational platform to a broader audience. This initiative democratizes access to interactive learning experiences, aligning with Sugar Labs' mission to provide free and open-source educational software. The accompanying documentation project will serve as a comprehensive resource, streamlining installation and fostering community engagement. By reaching the Raspberry Pi community, we stimulate the creation of educational content, encourage innovation, and cultivate a vibrant ecosystem of learners and educators. Overall, this launch amplifies Sugar Labs' impact, broadening its user base and empowering learners worldwide.

What technologies (programming languages, etc.) will you be using?

For the Raspberry Pi documentation project, the following technologies will be utilized:

Markdown: Used for writing and formatting documentation content.

Python: Utilized for scripting, automation, and development of code examples.

Bash Scripting: Used for automation of system tasks and setup procedures.

Git: Employed for version control and collaboration on documentation repositories.

Raspberry Pi OS (formerly Raspbian): The operating system for running Sugar and testing documentation procedures.

Sugar Desktop Environment: Explored for understanding and documenting educational activities.

GNU/Linux Command Line Tools: Utilized for system administration, package management, and troubleshooting on Raspberry Pi OS.

5.TimeLine:-

Break down the entire project into chunks and tell us what will you work on each week.

Weeks	Tasks to be Completed
Pre-GSOC (Apr - May)	<ul style="list-style-type: none"> • Test almost all the sugar activity and finalize a curated list of Sugar activities for Raspberry Pi. • Enhance my skills in packaging, Linux, and Bash scripting. • Deepen my understanding of the Sugar environment.
Community Bonding Period (1st May - 26th May)	<ul style="list-style-type: none"> • Collaborate with mentors to optimize and enhance Sugar activities. • Resolve all issues related to running Sugar on RPi across various environments. • Explore additional technologies and deepen my understanding of packaging and Sugar system.
Week-1 (27th May - June 2nd)	<ul style="list-style-type: none"> • I will resolve all issues related to the selected list, regardless of whether they are specific to RPi or not.
Week-2 (June 3rd - 9th)	<ul style="list-style-type: none"> • Port to Python 3 all the necessary Activities
Week-3 (June 10th -16th)	<ul style="list-style-type: none"> • Work on TurtleBlocks Activity
Week-4 (June 17th - 23rd)	<ul style="list-style-type: none"> • Work on Measure, Physics, Calculator Activity
Week-5 (June 24th - 30th)	<ul style="list-style-type: none"> • Work on Pippy Activity
Week-6 (July 1st - 7th)	<ul style="list-style-type: none"> • Midterm Evaluation of all the activities and work done till now <ul style="list-style-type: none"> • Work on Abacus Activity, Panorama Activity, Finance Activity, Countries Activity, Solar-System Activity, Ruler, Read activity.
Week-7 (July 8th - 14th) Midterm Evaluation	<ul style="list-style-type: none"> • Work on I can read activity, Color deducto activity, Connect the color

	activity, Fraction Bounce activity, Slide.
Week-8 (July 15th -21st)	<ul style="list-style-type: none"> • Work on Labyrinth activity, 2-Cars Activity, Pukklanapac, CardSort, Number-rush activity, Apple and Haken, Paths, Write activity.
Week-9 (July 22nd - 28th)	<ul style="list-style-type: none"> • Create two RPi specific activity/game, Number Guessing and Temperature Conversion Activity.
Week-10 (July 29th - Aug 4th)	<ul style="list-style-type: none"> • Making a Verbose documentation of everything in RPi-Docs
Week-11 (Aug 5th - Aug 11th)	<ul style="list-style-type: none"> • Making a Verbose documentation of everything in RPi-Docs <ul style="list-style-type: none"> • Packaging Sugar on RPi.
Week-12 (Aug 12th - Aug 18th)	<ul style="list-style-type: none"> • Packaging Sugar on RPi.
Final Evaluation	Final evaluation/testing and launch of sugar tailored for RPi environment.

6. Some Important Questions:-

How many hours will you spend each week on your project?

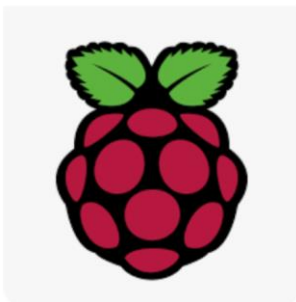
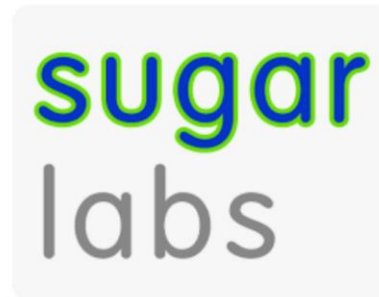
During my college summer vacation, commencing from May 15th to July 22nd, I anticipate being able to dedicate approximately 50-55 hours per week to GSoC-related activities. Prior to that period, I can allocate around 40-45 hours per week. With no other commitments during this time, I am fully prepared to devote most of my time to GSoC.

How will you report progress between evaluations?

I will maintain an active presence on GitHub by regularly submitting pull requests. Additionally, **I will actively engage on social media platforms such as LinkedIn, Instagram, and Twitter, where I will share updates on my progress, interact with the maker community, and promote participation in the project.**

Discuss your post-GSoC plans. Will you continue contributing to Sugar Labs after GSOC ends?

My primary post-GSoC plan involves making Sugar readily accessible within the maker community, attracting more developers and community members. This entails analysing additional activities and enhancing integration between Raspberry Pi and Sugar to ensure seamless functionality.



Corollary of all circuit diagrams used for setup:-

