FOSSASIA 2024 GSoC Proposal

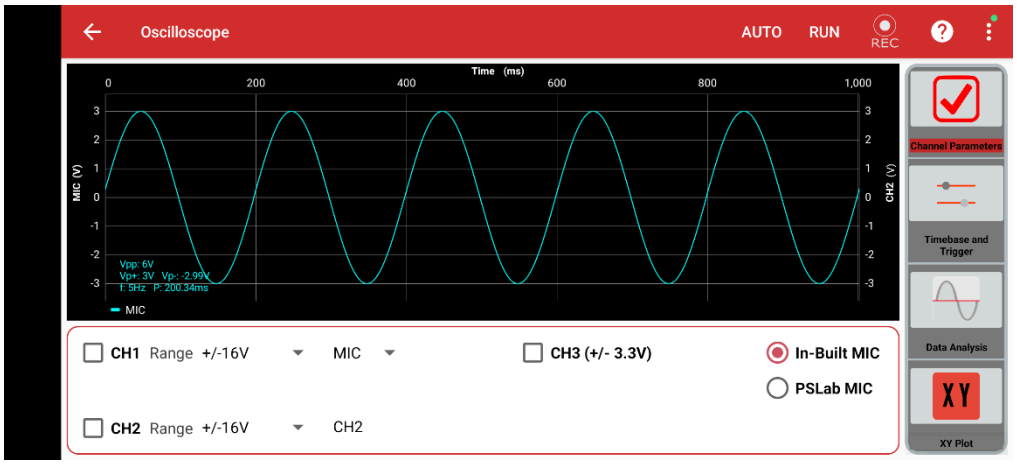
Update Oscilloscope in Android App of Pocket Science Lab and Implement Support for New Instruments

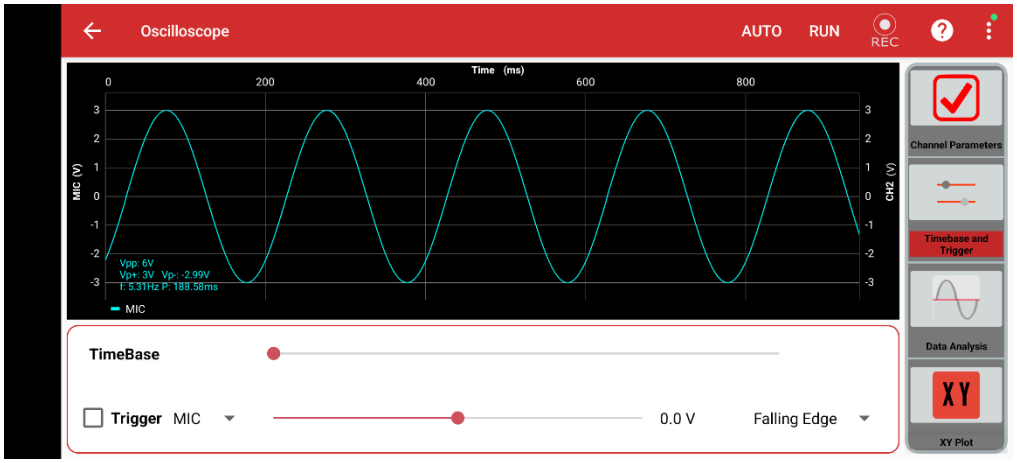
Anashuman Singh Cheema
GitHub: /AsCress
Gitter: @ascress:gitter.im
Email: anash.cheema@gmail.com

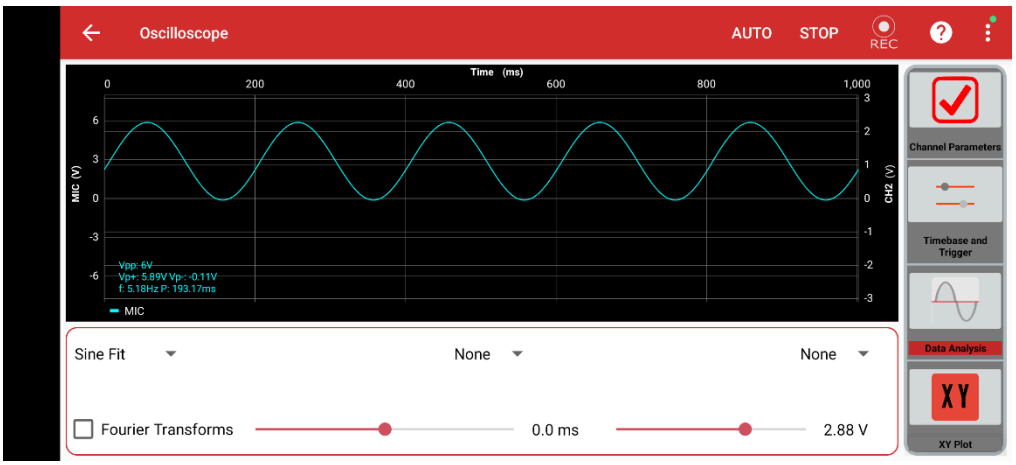
Table of Contents

Update Oscilloscope in Android App of Pocket Science Lab and Implement Support for New Instruments (*175* hours)

1. Updating the Oscilloscope:(A Revamped Experience)







Overview

The Oscilloscope has been one of the most useful and popular features of the PSLab Android application. With this project, my goal is to elevate its performance by integrating additional functionalities akin to those found in dedicated oscilloscope devices. Here are the proposed enhancements:

1. **Instrument Run Control**
   - 1.1 One of the most fundamental features which dedicated oscilloscope devices offer is the ability to Stop or Run the instrument.
   - 1.2 When the oscilloscope is running, it is acquiring data from the activated channels when the triggering conditions are met. When stopped, it displays the last acquired waveform.
   - 1.3 This allows for a detailed analysis of the instantaneously captured waveform.
2. **Horizontal & Vertical Position Controls**
   - 2.1 Another feature would be to control the horizontal and vertical position of the waveform on the oscilloscope display, for manual measurements.
   - 2.2 The horizontal position control moves the waveform back and forth on the display and can be used to either align the waveform with the horizontal divisions of the display or to view different sections of the waveform.
   - 2.3 The vertical position control moves the waveform up and down on the display and can be used for manual measurements in a similar way as the horizontal position control.
3. **Auto-Scale**
   - 3.1 This allows adjustment of the X-axis and Y-axis scale according to the acquired waveform in order to properly display it on the oscilloscope display.
   - 3.2 This feature would use algorithms to identify the waveform, measure its amplitude and period, and then scale the axes to properly display it.
4. **Triggering Controls**
   - 4.1 Currently, the oscilloscope supports only a single type of edge – triggering.
   - 4.2 All three-types of edge triggering: Rising-edge, Falling-edge and Dual-edge can be added.
5. **Automated Measurements**
   - 5.1 The oscilloscope as of now doesn't have a feature to automatically measure and display characteristics of the waveform such as amplitude, frequency, period, etc.
   - 5.2 This feature can be added which would calculate the characteristics of the waveform using algorithms and display them.
6. **Additional Features**

   The features mentioned above constitute only a small fraction of the extensive capabilities typically associated with an oscilloscope. I eagerly anticipate the opportunity to add more features beyond these initial offerings. These further enhancements can be explored through discussions with mentors and implemented as time allows, ensuring a more comprehensive and robust feature set for the application.

Why do I want to work on it?

The Pocket Science Lab application serves as an invaluable resource, granting access to a diverse array of laboratory instruments in situations where acquiring or affording them proves challenging. Personally, I've found it immensely beneficial for conducting experiments in non-traditional settings, such as at home or in environments lacking dedicated instruments like waveform generators and oscilloscopes.

However, I've noticed a gap in the functionality, particularly concerning the oscilloscope feature. While it allows for observation of signal waveforms, it lacks dedicated tools for precise measurements and waveform analysis.

In this project, my aim is to transform the oscilloscope from a mere observation tool into a comprehensive instrument capable of measuring various parameters of a signal. By enhancing these features, we can significantly augment the oscilloscope's utility, catering to the needs and preferences of the application's current user base.

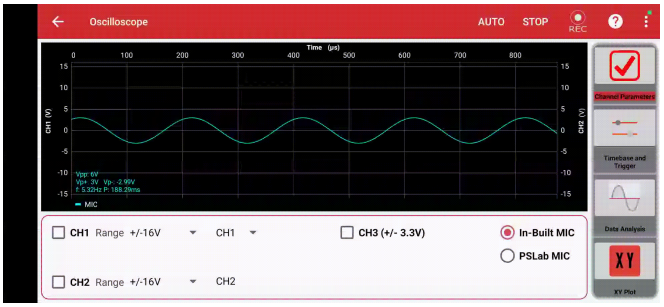
Implementing the Proposed Features

Instrument Run Control

This feature would allow the user to **Stop or Run** the oscilloscope. By stopping the oscilloscope, the user can capture a **single sequence** of the electrical signal.

As far the UI is concerned, it can be implemented as a **MenuItem** in the Toolbar of the OscilloscopeActivity (layout file – activity_landscape_menu.xml), as the Run/Stop button in conventional oscilloscopes is also given usually at the top.

This MenuItem would change its title to either “RUN” or “STOP”, indicating which mode the oscilloscope is currently in.



Demonstration showcasing the Run Control feature (click to view)

Horizontal and Vertical Position Controls

These controls would allow the user to **shift the waveform** up, down and move it back and forth on the display. These can be used to align the waveform with the horizontal and vertical divisions on the scales of the axes for **manual measurements**, or just to view the waveform properly on the display.

My initial thought was to add Crollers for adjusting the values, however keeping in mind the already designed UI and the user experience as the layout would scale to different screen sizes, **FloatSeekBars** are the best possible solution.

These would **adjust** their maximum and minimum values based on whatever would be the scale of the axes.

The **two implementations** are as follows, with the first one being my preferred choice: -

1. Adding the FloatSeekBars for horizontal and vertical position controls in the **DataAnalysisFragment** along with TextViews to display values.
2. Adding a whole **new fragment** for position controls.

Demonstration showcasing the Position Controls feature (click to view)

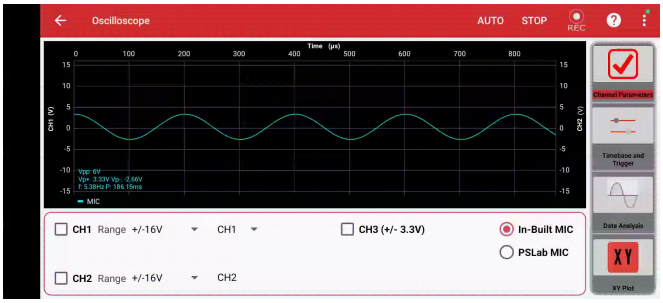
Auto-Scale

This feature would **adjust the scale** of both the axes appropriately with some padding to view the waveform properly on the display.

The button to Auto-Scale would be placed right beside the Run Control button as a **MenuItem** in the Toolbar of the OscilloscopeActivity.

When clicked, the acquired data would be analysed to find out **if a periodic waveform is present** or not. If only noise is detected, a Toast message saying “No waveform found”, would be displayed, else, the **period and amplitude of the wave will be calculated** and the axes would be scaled accordingly.

Whenever the scale of the axes is changed, the maximum and minimum values of the FloatSeekBars which control the position of the waveform will be changed, to facilitate the user for hassle-free measurements.



Demonstration showcasing the Auto-Scale feature (click to view)

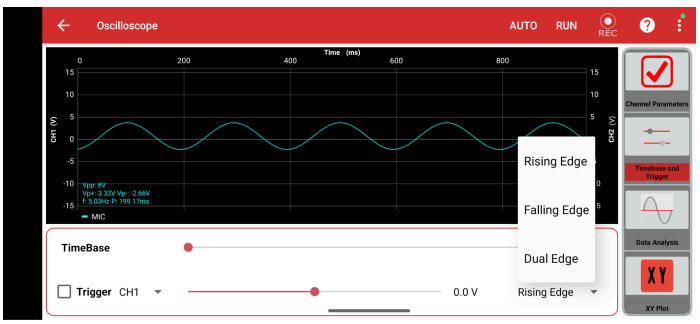
Triggering Controls

Although the oscilloscope already contains a single type of edge triggering, the triggering controls can be improved by introducing all three types of edge triggering: -

1. **Rising Edge Triggering** – The oscilloscope triggers the display of the waveform when the input signal crosses a predefined voltage threshold from a lower voltage level to a higher voltage level.
2. **Falling Edge Triggering** – The oscilloscope triggers the display of the waveform when the input signal crosses a predefined voltage threshold from a higher voltage level to a lower voltage level.
3. **Dual Edge Triggering** – The oscilloscope triggers the display of the waveform when the input signal crosses a predefined voltage threshold in either direction.

This can be implemented by adding a **Spinner** right in front of the trigger controls in the TimebaseTriggerFragment (layout file – fragment_timebase_tigger.xml), which would allow the user to switch between the three available options.

All three of them would be implemented programmatically in a similar way as the already available feature: -



Automated Measurements

One of the most needed features in a modern oscilloscope is the ability to **automatically measure** various characteristics of a signal. As supported by many conventional oscilloscopes, I propose the calculation and display of the following measurements of a signal: -

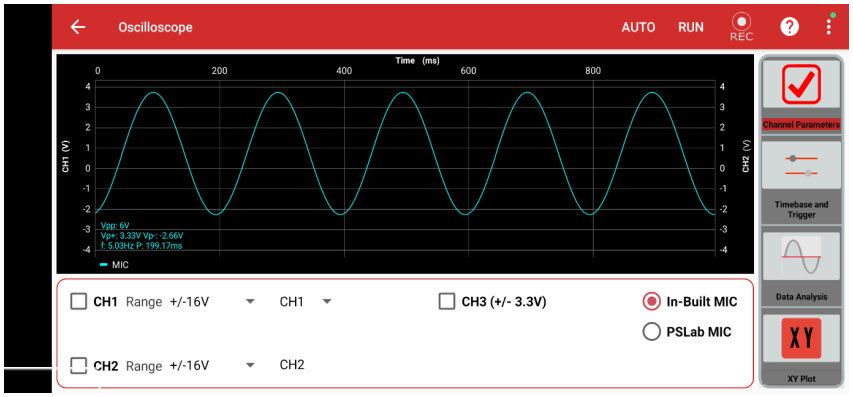
a. **Amplitude (Peak-to-peak) (V_{p-p})**
b. **Maximum Voltage (V_{max})**
c. **Minimum Voltage (V_{min})**
d. **Period**
e. **Frequency**

There are two possible choices for the display of these measurements, with the second one being my preferred one: -

1. Displaying these measurements in the **legend** of the graph.
2. Displaying these measurements in a **TextView** aligned towards the bottom left of the graph.

The user can have an option to **trigger the display** of these measurements on and off through the ToolBar menu.

Programmatically, these characteristics of a signal would be calculated using algorithms and if a **periodic waveform** is found, they would be displayed. In case only **noise** is detected, the values of these would be replaced by a '- ', as done by most of the available oscilloscopes.



Automated Measurements Display

Additional Features

Should there be any further enhancements planned for the application, a discussion can be held with the mentors, ensuring these features are seamlessly integrated within the designated timeframe.

Additionally, the app's user interface can undergo refinement to align with the latest Google Material Design Guidelines and Material Themes, thereby enhancing its overall aesthetic appeal and user experience.

2. Adding new Instruments in the App

Overview

Currently, the app has support for the following I2C compatible sensors: -

1. **ADS1115 – 4-channel 16-bit ADC**
2. **BMP180 – Temperature/Barometric Sensor**
3. **MLS90614 – IR Temperature Sensor**
4. **HMC5883L – 3-Axis Digital Compass IC**
5. **MPU6050 – Six Axis (Gyro + Accelerometer) Module**
6. **SHT21 – Digital Temperature and Humidity Sensor**
7. **TSL2561 – Light Sensor**
8. **MPU925X – MEMS Accelerometer, Gyroscope and Magnetometer**

I propose to add support for the following three sensors: -

1. **APDS9960 (Gesture, Color, Proximity, and Ambient Light Sensor): -**
   - The APDS9960 sensor detects gestures, measures color, proximity, and ambient light levels.
   - It is an I2C compatible sensor that requires simple commands for initialization and configuring the desired sensing modes.
2. **VL53L0X (Time-of-Flight Distance Sensor): -**
   - The VL53L0X sensor measures distance using time-of-flight principles.
   - It also communicates over I2C and can be initialized by configuring measurement settings and calibrating the sensor.
3. **CCS811 (Digital Gas Sensor): -**
   - The CCS811 is a digital gas sensor designed to detect a wide range if Volatile Organic Compounds (VOCs) and also includes a built-in relative humidity sensor.
   - It can be used to monitor indoor air quality (IAQ) with an I2C interface.

Furthermore, it's worth noting that while communication classes exist for various sensors like AD7718 and AD9833, a frontend UI has yet to be developed for them. This presents an additional opportunity for improvement within the scope of this project.

Why do I want to work on it?

The Pocket Science Lab application can be used to read data from a plethora of sensors, following the I2C standard. However, currently it supports only a few sensors, limiting its potential.

To address this, I propose the addition of new sensors, each offering unique insights and very different from one another.

In this project, my goal is to enhance the app by integrating support for a wide range of sensors within the allocated time frame. This objective can be further discussed with mentors for guidance and feasibility assessment.

Additionally, I aspire to enhance the user interface (UI) for all the sensors, aiming for a more intuitive and user-friendly experience. This aspect can also be addressed in discussions with mentors and implemented as time allows.

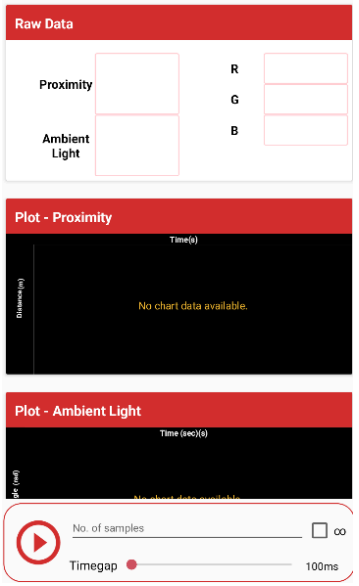
Implementing the Sensors

APDS9960 (Gesture, Color, Proximity and Ambient Light Sensor): -

The APDS9960 is a versatile digital, RGB, ambient light, proximity, and gesture sensor.

Some of its features: -

1. **RGB Color Sensing:** It can accurately measure red, green, and blue (RGB) light levels, allowing it to detect and differentiate between different colors in the environment.
2. **Ambient Light Sensing:** It includes an ambient light sensor capable of measuring the intensity of ambient light.
3. **Proximity Sensing:** It can detect the presence of objects or obstacles in close proximity to the sensor. It emits infrared (IR) light and measures the reflection to determine the distance between the sensor and nearby objects.
4. **Gesture Detection:** One of its standout features is its ability to recognize a variety of hand gestures, such as swipe, flick, and circle motions.

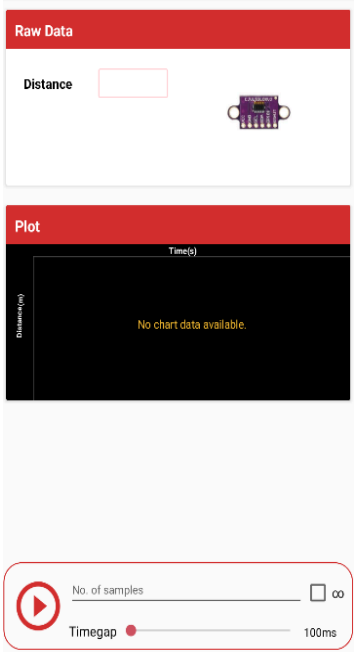


Mockup of the layout for SensorAPDS9960.java

VL53L0X (Time-of-Flight Distance Sensor): -

The VL53L0X is a time-of-flight (ToF) laser-ranging sensor module. It provides accurate distance measurement capabilities: -

1. **Laser-Based Ranging:** The VL53L0X utilizes a time-of-flight principle, where it emits short infrared laser pulses and measures the time it takes for the pulses to travel to an object and back, providing millimeter-level accuracy.
2. **Long Range Operation:** It offers a long-ranging capability with a maximum range of up to several meters, depending on the target surface characteristics and ambient lighting conditions.
3. **Fast Measurement Speed:** The sensor can perform distance measurements quickly, with typical measurement times in the range of a few milliseconds..
4. **Multi-Target Detection:** The VL53L0X is capable of detecting multiple targets within its field of view simultaneously. This feature is useful for applications such as occupancy detection, object tracking, and multi-touch interfaces.

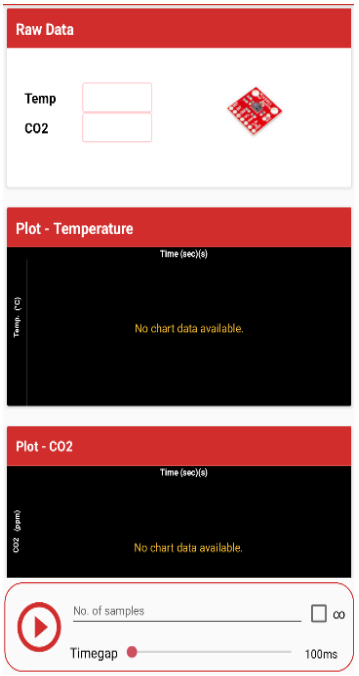


Mockup of the layout for SensorVL53L0X.java

CCS811 (Digital Gas Sensor): -

The CCS811 is an ultra-low-power digital gas sensor for monitoring indoor air quality (IAQ). It is designed to detect a wide range of volatile organic compounds (VOCs) and measure carbon dioxide (CO2) levels in the atmosphere. Here are some features:

1. **VOC Detection:** The CCS811 sensor can detect a wide variety of volatile organic compounds (VOCs), including alcohols, aldehydes, ketones, and organic acids.
2. **CO2 Sensing:** In addition to VOC detection, the CCS811 sensor is equipped with a built-in carbon dioxide (CO2) sensor. It can measure CO2 levels in the atmosphere.
3. **Dynamic Baseline Correction (DBC):** The sensor employs dynamic baseline correction algorithms to compensate for sensor drift over time. This ensures long-term stability and accuracy in VOC and CO2 measurements.
4. **On-Chip Processing:** The CCS811 sensor includes on-chip processing capabilities, allowing it to perform real-time data processing and output calibrated air quality measurements directly to the host microcontroller.



Mockup of the layout for SensorCCS811.java

Each of these enhancements can be seamlessly integrated following a standardized approach akin to the existing implementations. This involves the **creation of dedicated communication classes** for each sensor—such as APDS9960.java, VL53L0X.java, and CCS811.java—wherein methods are defined and calibration procedures are conducted.

Moreover, the user interface (UI) for these sensors will adhere to a consistent layout, presenting **raw data readings** and **graphical representations** in a clear and user-friendly manner.

Success Criteria

The project's success criteria are outlined as follows:

- Complete integration of all specified features aimed at enhancing the oscilloscope functionality within the app.
- Seamless functionality across all channels of the oscilloscope following the feature updates.
- Successful implementation and calibration of the three designated sensors within the application.
- Efficient integration and operation of all other I2C compatible sensors, involving the creation of layouts and activities tailored to their communication classes.
- Development of comprehensive unit testing protocols covering all newly implemented features throughout the project lifecycle.

Timeline

Phase	Time Period	Details	Est. Hours
Community Bonding Period	1 May - 26 May	Initial Preparation ● Finalizing the workflow of the project ● Devise the exact plans for feature implementations ● Decide on the UI implementations for all the features ● Familiarising myself with the relevant parts of PSLab Android codebase ● Finalize timeline and success criteria for project ● Finalize the draft UI/UX features of the project.	N/A
Phase 1	Week 1 27 May-2 June	Updating the Oscilloscope ● Implementing the Run Control Feature effectively. ● Creating user interface for Position Controls	20
	Week 2 3-9 June	Updating the Oscilloscope ● Developing Position Controls ● Implementing Auto Scale	20
	Week 3 10 June - 16 June	Updating the Oscilloscope ● Other related refactoring ● Buffer period for changes	10
	Week 4 17-23 June	Updating the Oscilloscope ● Implementing Trigger Controls ● Creating UI for Automated Measurements and designing	10
	Week 5 24-30 July	Updating the Oscilloscope ● Implementing Automated Measurements for all channels ● Looking for errors in measured values and calibration	5 5
	Week 6 31-7 July	Updating the Oscilloscope ● Other related refactoring Buffer Period for unforeseen changes	
Week 6 8-12 July		**Midterm Evaluations**	
Phase 2	Week 7 13-19 July	Adding new Instruments ● Creating the user interface for all the sensors to be integrated	20

		• Creating the Java communication class for the APDS9960 sensor	
	Week 8 20-26 July	Adding new Instruments • Developing activity for and finalizing the APDS9960 sensor • Creating the Java communication class for VL53L0X	20
	Week 9 27 July-2 Aug	Adding new Instruments • Developing activity for and finalizing the VL53L0X sensor • Creating the Java communication class for CCS811	20
	Week 10 3-9 Aug	Adding new Instruments • Developing activity for and finalizing the CCS811 sensor • Testing all the sensors and calibration	20
	Week 11 10-19 Aug	Adding new Instruments • Creating activities and UI for all the remaining sensors. • Add unit tests	20
	Week 12 20 Aug – 26 Aug	Buffer Period for unforeseen changes	10
Week 13 26-2 Sep		**Submit Final Evaluations**	

About Me

Personal Information

- **Anashuman Singh Cheema**
- **E-mail :** anash.cheema@gmail.com
- **GitHub:** /AsCress
- **Contact No.:** +91-9810515042

- **Studying:** B.Tech Electronics and Communication Engineering (2nd year) at Netaji Subhas University of Technology, Dwarka, New Delhi
- **Technical Skills:** C, C++, Python, HTML5, CSS, JavaScript, Java (Android), XML, Kotlin, Jetpack Compose, SQL
- **Time Zone:** Indian Standard Time (GMT +5:30)

Hey there ! My name is Anashuman Singh Cheema. I am currently pursuing B.Tech in Electronics and Communication Engineering (ECE) at Netaji Subhas University of Technology (NSUT), Dwarka, New Delhi.

Currently, I hold the role of Senior App Developer at Google Developer Student Club (GDSC), NSUT, while also serving as a Mentor within the Algorithm Society of NSUT (ASN).

I developed a passion for coding at a young age, initially diving into C++ as my inaugural programming language. From there, my exploration extended across various domains, such as Web and App development. My fascination with App development deepened over time, leading me to undertake numerous projects even before joining college.

Upon entering college, I began actively participating in open-source projects alongside my personal endeavors. It was during this time that I first engaged in collaboration with fellow developers, gaining firsthand experience in learning while actively developing.

The open-source community has given me so much till now, from imparting the skills to navigate and comprehend extensive codebases to fostering collaborations with other developers, thereby instilling a great deal of confidence along the way. Contributing to the open-source community is a prospect I eagerly anticipate in the future.

All the work, that I have done till now on the PSLab Android App is available here:-

https://github.com/AsCress/pslab-android

Here, is information about some of my other projects:-

1. ParkIN(Android):- Developed an app that suggests the users off-street parking around their current location.

   https://github.com/AsCress/ParkIN

2. Money Mentor(Android):- Developed an app that helps users manage finances, along with an AI bot for personalized suggestions.

   https://github.com/AsCress/MoneyMentor