SkyWatch

By Samuel, Alex & Marcus

A Day in the Life...

What SkyWatch accomplishes

1. **Collects weather data using Pi components**
2. **Active processing on Pi - No external computing**
3. **Plug and use functionality - Easy to set up**
4. **Access info from the app - Convenient**

Why Choose Us?

Traditional Weather Systems

- Managed by large corporations
- Non transparent
- No real time data display

Personal weather stations

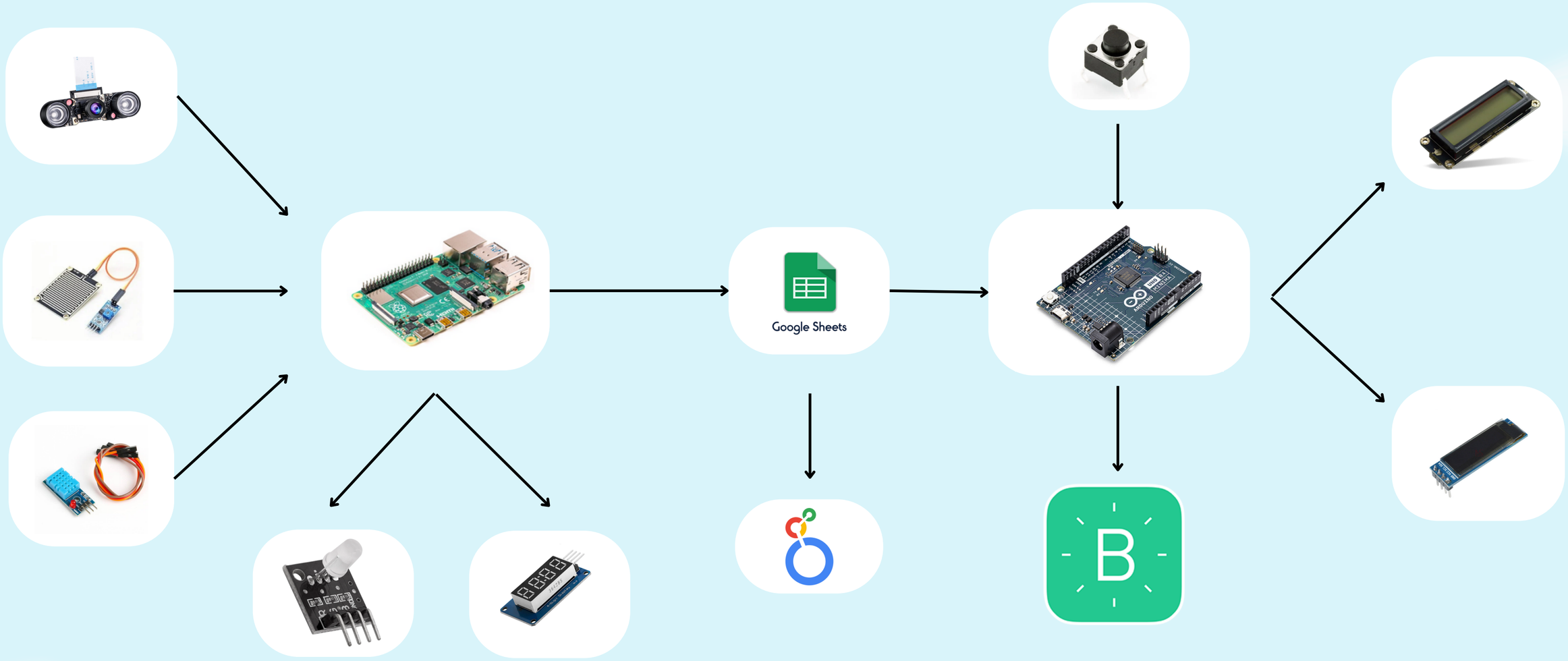
- Expensive
- Hard to configure
- Semi transparent

SkyWatch

- Cheap
- Small company owned
- Completely open source
- Easy to set up
- Up-to-date data

Making SkyWatch

Block Diagram



How it Works- Input

1.

Pi Camera

Takes images of the clouds.

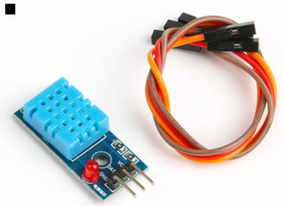
2.

Raindrop Sensor

Detects:

- Moisture
- Precipitation

3.



DHT 11 Sensor

Detects:

- Humidity
- Temperature

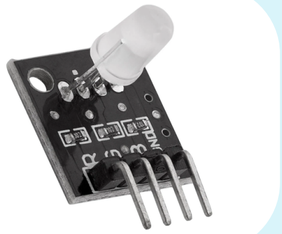
Data collection takes place every 5 minutes.

The data collected is then uploaded onto Google Sheets via HTTPS.

How it works- Raspberry Pi

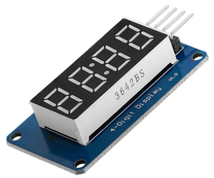
App Script

Configures and automates entries



LED

Turns on when data is getting collected



8 Seg-LED Display

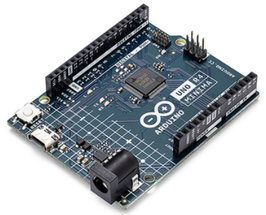
Displays timer before next data collection

Google Sheets

Pi sends the data to Google Sheets via HTTPS requests

How it works- Google Sheets

Google Sheets

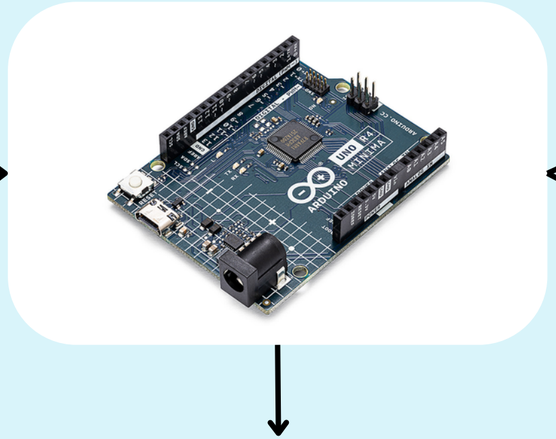


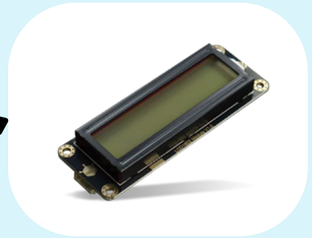
Arduino pulls data from database,
sending data to Blynk studio for visualization

Looker Dashboard pulls data from database

How it works- Arduino

Button Input



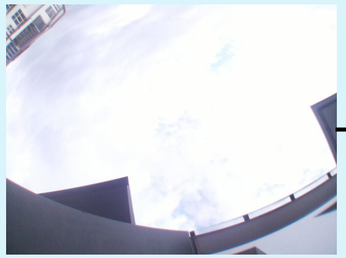


OLED and LCD Display

Button allows for instantaneous data collection!

Making More Data

Input image:

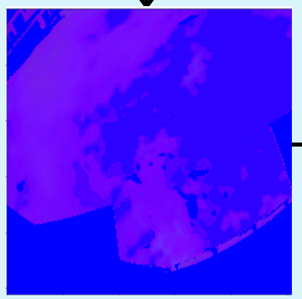


Resize & use K-means clustering Create mask to see the sky only.

Binary Mask

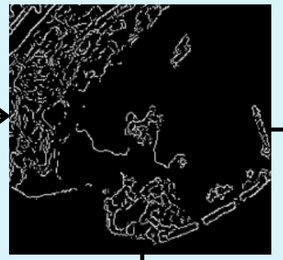
Cloud % data

Convert image to HSV

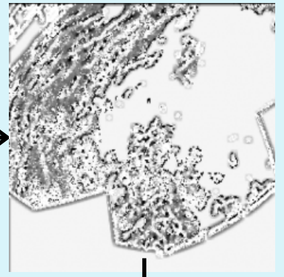


Hue, Saturation and Value mean data

Grayscale, Use edge Detection & Local Binary Pattern



Edge Count data



Local Binary Pattern- mean, STD, Entropy

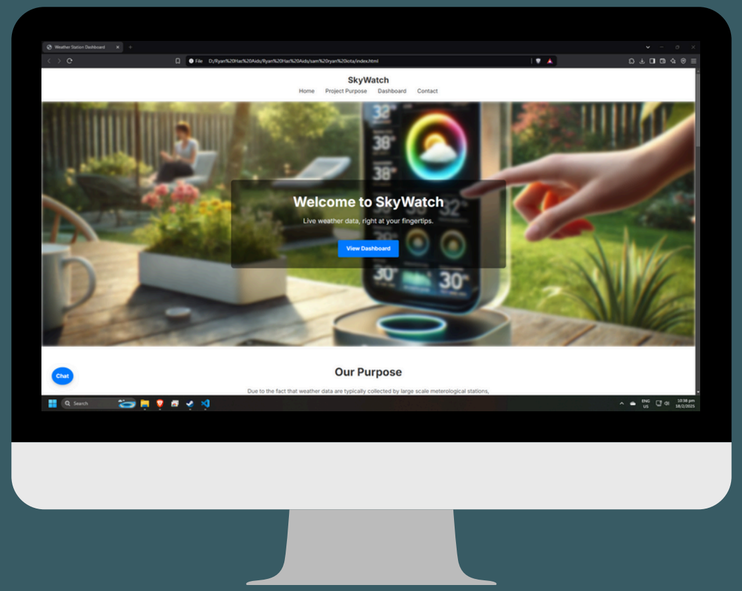
How We Predict Rain?

```
function checkRainPrediction(rowData) {
    // Define weights
    var weights = {
        temperature: 0.15,
        humidity: 0.25,
        edgeCount: 0.15,
        lbpMean: 0.1,
        lbpStdDev: 0.05,
        lbpEntropy: 0.05,
        hueMean: 0.05,
        saturationMean: 0.05,
        valueMean: 0.05,
        cloudPercentage: 0.2
    };
```

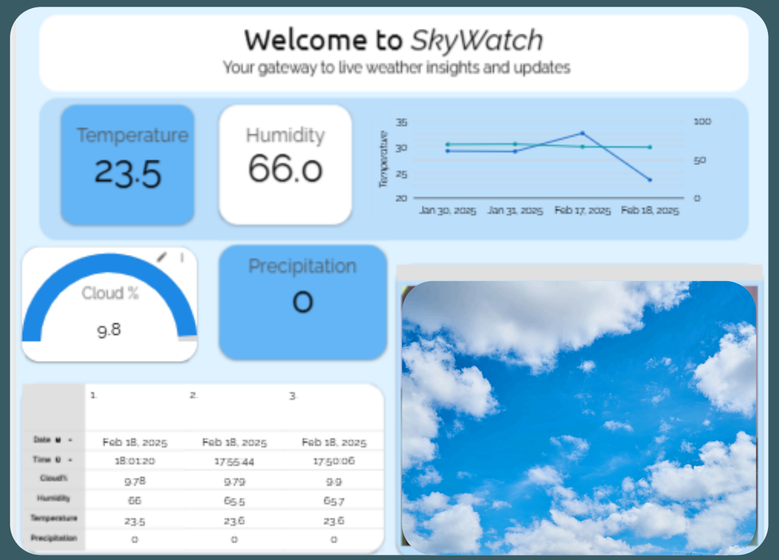
By summing the weighted values and comparing it with the threshold, we can predict if it will rain.

The Final Results...

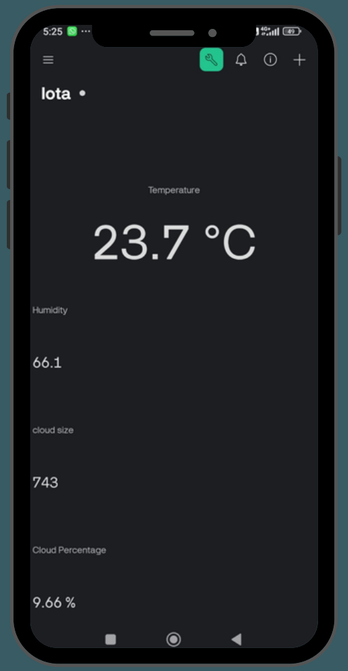
Website



Looker Studio Dashboard



Blynk App



Future Implementations

Groundwork for Machine Learning

Use machine learning to:

- Make predictions more reliable
- Make app more insightful
- Make more critical observations

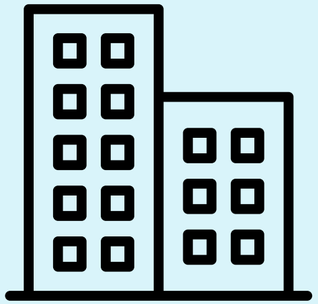
```
#HSV Image
masked_hsv_image = cv2.cvtColor(masking, cv2.COLOR_BGR2HSV)
#Grayscale Image
masked_gray_image = cv2.cvtColor(masking,cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(20, 10))

plt.subplot(1, 2, 1)
plt.title("HSV")
plt.imshow(masked_hsv_image)

plt.subplot(1, 2, 2)
plt.title("Gray")
plt.imshow(masked_gray_image, cmap="gray")
#Smooth out:
masked_blurred_image= cv2.GaussianBlur(masked_gray_image, (11, 11), 0)
plt.imshow(masked_blurred_image, cmap ="gray")
#Edge data:
masked_edges = cv2.Canny(masked_blurred_image, 100, 200)
plt.imshow(masked_edges, cmap = 'gray')
edge_count = np.sum(masked_edges > 0)
#LBP Data:
from skimage.feature import local_binary_pattern
radius = 3  # Radius of the circular LBP
n_points = 8 * radius  # Number of points to consider for the circular LBP
lbp_image = local_binary_pattern(masked_gray_image, n_points, radius, method='uniform')
lbp_hist, lbp_bins = np.histogram(lbp_image.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
lbp_hist_normalized = lbp_hist / lbp_hist.sum()
#HSV Data
cloud_mask = np.where((masked_hsv_image[..., 2] > 100), 1, 0)
cloud_pixels = masked_hsv_image[cloud_mask == 1]
avg_hue = np.mean(cloud_pixels[..., 0])  # Hue
avg_saturation = np.mean(cloud_pixels[..., 1])  # Saturation
avg_value = np.mean(cloud_pixels[..., 2])  # Brightness
print(avg_hue, avg_saturation, avg_value)
#Cloud% data:
cloud_pixels = mask.size # Count cloud pixels
total_pixels = masking_flat.size  # Total pixels in the image
```


Demo

How Our Project Can Help You

01 Personal Use

Get real time weather updates before heading out



02 Companies and Corporations

Enables efficient and reliable tracking for planning outdoor events

03 Community & Smart Cities

Provide real-time local weather data for neighborhood awareness.