DAFTAR PUSTAKA

Ahmad, Y. A., Surya Gunawan, T., Mansor, H., Hamida, B. A., Fikri Hishamudin, A., & Arifin, F. (2021). On the Evaluation of DHT22 Temperature Sensor for IoT Application. *Proceedings of the 8th International Conference on Computer and Communication Engineering, ICCCE 2021*, 131–134. https://doi.org/10.1109/ICCCE50029.2021.9467147

Akila, I. S., Sivakumar, A., & Swaminathan, S. (2017). Automation in plant growth monitoring using high-precision image classification and virtual height measurement techniques. *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, *2018-January*, 1–4. https://doi.org/10.1109/ICIIECS.2017.8275862

Alqinsi, P., Matheus Edward, I. J., Ismail, N., & Darmalaksana, W. (2018). IoT-Based UPS Monitoring System Using MQTT Protocols. 2018 4th International Conference on Wireless and Telematics (ICWT), 1–5. https://doi.org/10.1109/ICWT.2018.8527815

Amalia, F. (2020). Perancangan Sistem Berbasis Internet of Things (IoT) untuk Efisiensi Biaya Pemakaian Energi Listrik pada Gedung Kuliah Jurusan Teknik Elektro.

Begini Cara Hitung Besaran kWh yang Diperoleh dari Setiap Pembelian Token Listrik PLN. (n.d.). Retrieved July 19, 2023, from https://web.pln.co.id/media/siaran-pers/2022/02/begini-cara-hitung-besaran-kwh-yang-diperoleh-dari-setiap-pembelian-token-listrik-pln

Bhatt, D.; Patel, C.; Talsania, H.; Patel, J.; Vaghela, R.; Pandya, S.; Modi, K.; Ghayvat, H. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. Electronics 2021, 10, 2470. https://doi.org/10.3390/ electronics10202470

Cao, G., Russell, R. M., Lischner, N., & Prior, R. L. (1998). Serum Antioxidant Capacity Is Increased by Consumption of Strawberries, Spinach, Red Wine or Vitamin C in Elderly Women. The Journal of Nutrition, 128(12), 2383–2390. https://doi.org/10.1093/JN/128.12.2383

DFRobot. (2020). *Gravity: Analog TDS Sensor , Meter For Arduino SKU SEN0244-DFRobot*. 1. https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244

Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. *Informatics 2021, Vol. 8, Page 79*, *8*(4), 79. https://doi.org/10.3390/INFORMATICS8040079

Eridani, D., Wardhani, O., & Widianto, E. D. (2017). Designing and implementing the arduino-based nutrition feeding automation system of a prototype scaled nutrient film technique (NFT) hydroponics using total dissolved solids (TDS) sensor. Proceedings - 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2017, 2018-January(October), 170–175. https://doi.org/10.1109/ICITACEE.2017.8257697

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, Upper Saddle River.

Gravity__Analog_pH_Sensor_Meter_Kit_V2_SKU_SEN0161-V2-DFRobot. (n.d.). Retrieved July 14, 2023, from https://wiki.dfrobot.com/Gravity__Analog_pH_Sensor_Meter_Kit_V2_SKU_SEN0161-V2#target_6

Hartono, R. N. F. (2015). Pengaruh Konsentrasi Nutrisi Pada Pertumbuhan Tanaman Bayam Cabut (Amaranthus Tricolorl.) Secara Hidroponik Pada Greenhouse (Doctoral dissertation, Universitas Gadjah Mada).

Hidayanti, L., & Kartika, T. (2019). Pengaruh Nutrisi AB Mix Terhadap Pertumbuhan Tanaman Bayam Merah (Amaranthus tricolor L.) secara Hidroponik. Sainmatika: Jurnal Ilmiah Matematika Dan Ilmu Pengetahuan Alam, 16(2). https://doi.org/10.31851/sainmatika.v16i2.3214

Himeur, Y., Alsalemi, A., Bensaali, F., & Amira, A. (2021). Smart power consumption abnormality detection in buildings using micromoments and improved K-nearest neighbors. *International Journal of Intelligent Systems*, *36*(6), 2865–2894. https://doi.org/10.1002/INT.22404

Ichwana, Nasution, I. S., Sundari, S., & Rifky, N. (2020). Data Acquisition of Multiple Sensors in Greenhouse Using Arduino Platform. IOP Conference Series: Earth and Environmental Science, 515(1). https://doi.org/10.1088/1755-1315/515/1/012011

Irawan, Y., Febriani, A., Wahyuni, R., & Devis, Y. (2021). Water quality measurement and filtering tools using Arduino Uno, PH sensor and TDS meter sensor. Journal of Robotics and Control (JRC), 2(5), 357–362. https://doi.org/10.18196/jrc.25107

Istiqomah, S. (2007). *Menanam Hidroponik*. Ganeca Exact.

John, M. (2018). Comparative study on various system based on Raspberry-Pi Technology. *International Research Journal of Engineering and Technology (IRJET)*, *5*(01), 1486-1488.

Karamina, H., Fikrinda, W., & Murti, A. T. (2017). Kompleksitas pengaruh temperatur dan kelembaban tanah terhadap nilai pH tanah di perkebunan jambu biji varietas kristal (Psidium guajava l.) Bumiaji, Kota Batu. Kultivasi, 16(3). http://journal.unpad.ac.id/kultivasi/article/view/13225

Khan, Z. A., Chattha, S. H., & Shaikh, I. (2021). Comparative Assessment of Hydroponic and Geoponic Cultivation Systems for Sustainable Spinach Cultivation. https://doi.org/10.17582/journal.pjar/2021/34.4.678.688

Khudoyberdiev, A.; Ahmad, S.; Ullah, I.; Kim, D. An Optimization Scheme Based on Fuzzy Logic Control for Efficient Energy Consumption in Hydroponics Environment. *Energies* **2020**, *13*, 289. https://doi.org/10.3390/en13020289

Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., & Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. Computer Science Review, 40. https://doi.org/10.1016/J.COSREV.2021.100364

Lestari G. 2009. Berkebun Sayuran Hidroponik di Rumah. Jakarta : Prima Info Sarana.

Liao, Y., & Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. Computers & security, 21(5), 439-448.

Liu, S., Kuang, H., & Lai, Z. (2014). Transcriptome Analysis by Illumina High-Throughout Paired-End Sequencing Reveals the Complexity of Differential Gene Expression during In Vitro Plantlet Growth and Flowering in Amaranthus tricolor L. PLOS ONE, 9(6), e100919. https://doi.org/10.1371/JOURNAL.PONE.0100919

Liu, T. (2015). Digital-output relative humidity & temperature sensor/module DHT22. *New York : Aosong Electronic*, *22*, 1–10. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

Liu, X., Deng, Z. & Yang, Y. Recent progress in semantic image segmentation. *Artif Intell Rev* 52, 1089–1106 (2019). https://doi.org/10.1007/s10462-018-9641-3

M. Hosny, K., Magdi, A., Salah, A., El-Komy, O., & Lashin, N. A. (2023). Internet of things applications using Raspberry-Pi: a survey. International Journal of Electrical and Computer Engineering (IJECE), 13(1), 902. https://doi.org/10.11591/ijece.v13i1.pp902-910

Manual, P. R. (2022). Arduino UNO R3 Features. *Https://Docs.Arduino.Cc*, 1–13. https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Mucherino, A., Papajorgji, P. J., & Pardalos, P. M. (2009). *k-Nearest Neighbor Classification*. 83–106. https://doi.org/10.1007/978-0-387-88615-2_4

Nugraha, R. U., & Susila, A. D. (2015). *Sumber sebagai hara pengganti AB mix pada budidaya sayuran daun secara hidroponik.* Jurnal Hortikultura Indonesia, 6(1), 11-19.

Nugroho, K. S. (2019). Confusion Matrix untuk Evaluasi Model pada Supervised Learning. Medium. https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f

Nuswantoro, A. (2015). Perbaikan dan Penggantian Sistem Pendingin Mesin Opel Blazer DOHC LT" Perbaikan Kebocoran Pada Radiator".

Ribera, A., Bai, Y., Wolters, A. M. A., van Treuren, R., & Kik, C. (2020). A review on the genetic resources, domestication and breeding history of spinach (Spinacia oleracea L.). Euphytica, 216(3), 1–21. https://doi.org/10.1007/S10681-020-02585-Y/TABLES/3

Rs-Components. (2019). Datasheet Raspberry Pi Model B. *Raspberrypi.Org*, *June*, 1. https://datasheets.raspberrypi.org

Rukmana, Rahmat. (1994). *Bayam : bertanam dan pengolahan pascapanen / Rahmat Rukmana*. Yogyakarta :: Kanisius,.

Saha, S., Kabir, S., & Rajib, R. H. (2018). IoT Based Automated Fish Farm Aquaculture Monitoring System Digital Evaluation of Broad Question Answer Script View project A Keyword Based Technique to Evaluate Broad Question Answer Script View project IoT Based Automated Fish Farm Aquaculture Monitoring System. 27–28. https://doi.org/10.1109/ICISET.2018.8745543

Sani, H. A., Rahmat, A., Phd, M. I., Phd, R. R., & Endrini, S. (2004). Potential anticancer effect of red spinach (Amaranthus gangeticus) extract. Asia Pac J Clin Nutr, 13(4), 396–400.

Saputra, A. H., & Fudholi, D. H. (2021). Realtime Object Detection Masa Siap Panen Tanaman Sayuran Berbasis Mobile Android Dengan Deep Learning. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 5(4). https://doi.org/10.29207/resti.v5i4.3190

Saputri, D. W. (2024). Skripsi Implementasi Soft Voting Classifier Untuk Prediksi Kinerja Mitra (Studi Kasus : Pt Telkom Indonesia).

Soni, D., & Makwana, A. (2017). *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT) Analysis and Survey on String Matching Algorithms for Ontology Matching View project MP-Index View project A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)*. https://www.researchgate.net/publication/316018571

Srinidhi, H. K., Shreenidhi, H. S., & Vishnu, G. S. (2020). Smart Hydroponics system integrating with IoT and Machine learning algorithm. Proceedings -

5th IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2020, 261–264. https://doi.org/10.1109/RTEICT49044.2020.9315549

Tintondp, 2016. *Hidroponik Wick System (Cetakan ke-2)*. Jakarta : PT. Agromedia.

Wahid, F., & Kim, D. (2016). A prediction approach for demand analysis of energy consumption using k-nearest neighbor in residential buildings. International Journal of Smart Home, 10(2), 97-108.

Xu, Shuyuan; Wang, Jun; Shou, Wenchi; Ngo, Tuan; Sadick, Abdul-Manan; Wang, Xiangyu (2020). *Computer Vision Techniques in Construction: A Critical Review. Archives of Computational Methods in Engineering, (), –.* doi:10.1007/s11831-020-09504-3

Yong, Y. Y., Dykes, G., Lee, S. M., & Choo, W. S. (2016). Comparative Study of Betacyanin Profile and Antimicrobial Activity of Red Pitahaya (Hylocereus polyrhizus) and Red Spinach (Amaranthus dubius). Plant Foods for Human Nutrition 2016 72:1, 72(1), 41–47. https://doi.org/10.1007/S11130-016-0586-X

Lampiran 1. Python Notebook untuk Pembuatan Model Machine Learning pada Tingkat Pertumbuhan Bayam Hijau


```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def segment_image(image_array):
    # Tentukan rentang warna yang ingin disegmentasi dalam format RGB
    color_ranges = [
        # Rentang warna minggu 2
        (np.array([16, 83, 1]), np.array([107, 133, 86])),
        (np.array([21, 61, 0]), np.array([64, 147, 56])),
        (np.array([112, 156, 0]), np.array([151, 205, 10])),
        # Rentang warna minggu 3
        (np.array([28, 114, 29]), np.array([72, 182, 83])),
        (np.array([48, 193, 106]), np.array([80, 205, 161])),
        # Rentang warna minggu 4
        (np.array([92, 207, 47]), np.array([240, 247, 239])),
        (np.array([57, 145, 15]), np.array([157, 246, 116])),
        (np.array([12, 166, 102]), np.array([40, 226, 151])),
        (np.array([5, 113, 71]), np.array([34, 175, 126])),
        (np.array([0, 121, 73]), np.array([8, 144, 124])),
        (np.array([0, 110, 66]), np.array([14, 138, 69])),
        (np.array([1, 105, 58]), np.array([18, 118, 79])),
        (np.array([27, 237, 187]), np.array([142, 243, 233])),
        (np.array([0, 167, 132]), np.array([9, 193, 157])),
        (np.array([0, 87, 55]), np.array([3, 111, 71])),
        (np.array([0, 129, 108]), np.array([14, 199, 153]))
    ]

     # Rentang warna yang ingin dihindari dalam format RGB
    avoid_ranges = [
        # Rentang warna minggu 2
        (np.array([3, 180, 186]), np.array([73, 215, 217])),
        (np.array([54, 112, 77]), np.array([70, 133, 99])),
        (np.array([57, 106, 81]), np.array([74, 125, 92])),
        (np.array([40, 92, 75]), np.array([74, 130, 101])),
        (np.array([23, 93, 79]), np.array([47, 104, 87])),
        (np.array([69, 112, 69]), np.array([82, 125, 81])),
        (np.array([30, 85, 80]), np.array([40, 90, 90])),
        (np.array([96, 128, 82]), np.array([116, 149, 93])),
        (np.array([62, 100, 76]), np.array([106, 138, 93])),
        (np.array([183, 234, 168]), np.array([191, 241, 186])),
        (np.array([110, 137, 79]), np.array([187, 212, 103])),
        (np.array([137, 154, 79]), np.array([197, 201, 130])),
        (np.array([103, 211, 197]), np.array([135, 241, 232])),
        (np.array([43, 87, 66]), np.array([78, 100, 78])),
        (np.array([18, 83, 79]), np.array([28, 90, 86])),
        (np.array([85, 87, 32]), np.array([95, 115, 66])),
        (np.array([20, 78, 71]), np.array([41, 93, 79])),
        (np.array([86, 129, 89]), np.array([115, 159, 101])),
        (np.array([121, 202, 158]), np.array([127, 214, 182])),
        (np.array([46, 83, 52]), np.array([111, 126, 65])),
        (np.array([31, 82, 64]), np.array([73, 115, 90])),
        (np.array([146, 222, 177]), np.array([198, 243, 233])),
        (np.array([70, 232, 231]), np.array([111, 245, 243])),
        (np.array([86, 124, 67]), np.array([117, 162, 93])),
        (np.array([9, 83, 72]), np.array([48, 106, 83])),
        # Rentang warna minggu 3
        (np.array([27, 66, 37]), np.array([40, 83, 53])),
        (np.array([20, 65, 45]), np.array([32, 81, 63])),
        (np.array([25, 67, 47]), np.array([52, 92, 65])),
        (np.array([240, 242, 239]), np.array([240, 242, 239]))

    ]

    # Membuat masker warna untuk rentang warna yang ingin dihindari
    avoid_color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in avoid_ranges:
        avoid_color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Membuat masker warna untuk rentang warna yang ingin disegmentasi
```

```
    color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in color_ranges:
        color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Menghilangkan area yang ingin dihindari dari masker warna hasil segmentasi
    color_mask &= ~avoid_color_mask

    return color_mask


def show_image(image_array,color_mask):
    # Buat gambar hasil segmentasi
    segmented_array = np.zeros_like(image_array)
    segmented_array[color_mask] = image_array[color_mask]

    # Tampilkan gambar asli dan gambar yang telah disegmentasi
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image_array)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(segmented_array)
    axes[1].set_title('Segmented Image')
    axes[1].axis('off')

    plt.show()


def upload_image(filename):
    try:
        image = Image.open(filename)
        image_array = np.array(image)
        color_mask = segment_image(image_array)
        return image_array, color_mask
    except Exception as e:
        print(e)
        return None, None


test = [
    '/content/drive/MyDrive/bayam hijau/minggu 2/2023-08-28_13_19_38.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 3/2023-09-06_12_31_19.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-13_00_45_18.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 3/2023-09-08_15_38_22.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 2/2023-09-02_14_39_23.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 2/2023-08-29_01_21_31.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-13_03_46_40.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-15_10_17_44.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 2/2023-09-05_10_14_36.jpg',
    '/content/drive/MyDrive/bayam hijau/minggu 2/2023-09-04_06_54_48.jpg'
]

for loop in test:
  image_array, color_mask = upload_image(loop)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(loop)
  show_image(image_array, color_mask)
```

184210
/content/drive/MyDrive/bayam hijau/minggu 2/2023-08-28_13_19_38.jpg

Original Image

Segmented Image

367390
/content/drive/MyDrive/bayam hijau/minggu 3/2023-09-06 12 31 19.jpg

Original Image

Segmented Image

594381
/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-13_00_45_18.jpg

Original Image

Segmented Image

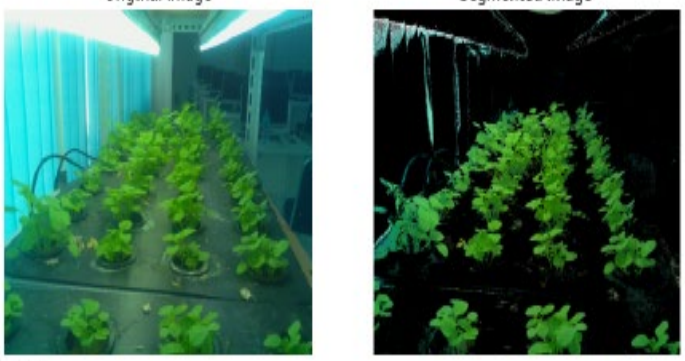
411910
/content/drive/MyDrive/bayam hijau/minggu 3/2023-09-08_15_38_22.jpg

Original Image

Segmented Image

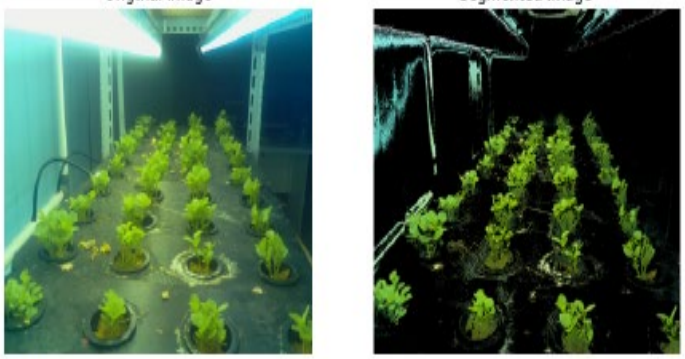
312860
/content/drive/MyDrive/bayam hijau/minggu 2/2023-09-02 14 39 23.jpg

Original Image Segmented Image



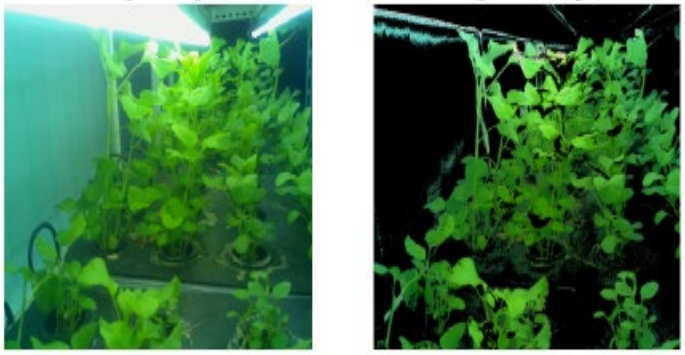
259912
/content/drive/MyDrive/bayam hijau/minggu 2/2023-08-29 01 21 31.jpg

Original Image Segmented Image



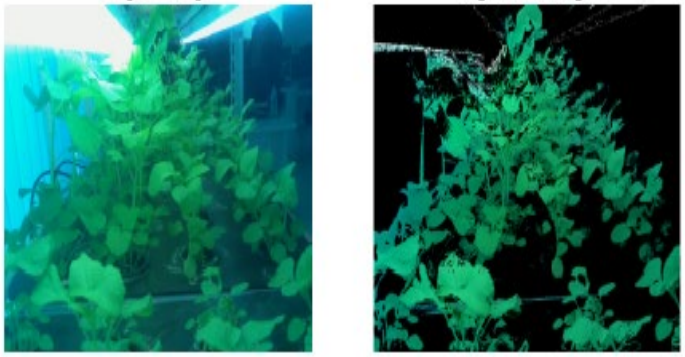
576074
/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-13_03_46_40.jpg

Original Image Segmented Image



506229
/content/drive/MyDrive/bayam hijau/minggu 4/2023-09-15_10_17_44.jpg

Original Image Segmented Image



380756
/content/drive/MyDrive/bayam hijau/minggu 2/2023-09-05_10_14_36.jpg

Original Image Segmented Image

```
#evaluate the algorithm to classify

def classify_image(filename):
    _, color_mask = upload_image(filename)
    if color_mask is None:
      return "wrong data"
    num_pixels = np.sum(color_mask)

    if num_pixels <= 360000:
      return (0, num_pixels, filename)
    elif num_pixels <= 500000:
      return (1, num_pixels, filename)
    else:
      return (2, num_pixels, filename)
```



```
image_pixels = []
image_groups = []
predict_groups = []

import os

for image in os.listdir("/content/drive/MyDrive/bayam hijau/minggu 2"):
    results = classify_image("/content/drive/MyDrive/bayam hijau/minggu 2/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(0)
    predict_groups.append(results[0])


for image in os.listdir("/content/drive/MyDrive/bayam hijau/minggu 3"):
    results = classify_image("/content/drive/MyDrive/bayam hijau/minggu 3/"+image)
    if results == "wrong data":
      continue

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(1)
    predict_groups.append(results[0])

for image in os.listdir("/content/drive/MyDrive/bayam hijau/minggu 4"):
    results = classify_image("/content/drive/MyDrive/bayam hijau/minggu 4/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(2)
    predict_groups.append(results[0])
```

```
cannot identify image file '/content/drive/MyDrive/bayam hijau/minggu 3/2023-09-06_12_49_33.jpg'
```

```
# Import necessary libraries
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt


print(confusion_matrix(image_groups, predict_groups))
print(f"Accuracy (might need rearrangement): {accuracy_score(image_groups, predict_groups)}")
```

```
[[180  14   0]
 [  0  23  12]
 [  0  17  47]]
Accuracy (might need rearrangement): 0.8532423208191127
```

```
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

# Split the data into training and test sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(image_pixel_values, image_groups, test_size=0.3, random_state=42)

# Create a random forest classifier and train it
clf = RandomForestClassifier(n_estimators=200, random_state=0)
clf.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = clf.predict(X_test)

# Print the accuracy and classification report
print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")
```

```
Accuracy: 87.50%
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[61  1  0]
 [ 0  5  4]
 [ 0  6 11]]
```

```
# save the model
import joblib
joblib.dump(clf, 'bayam_putih.pkl')
```

```
['bayam_putih.pkl']
```

Lampiran 2. Python Notebook untuk Pembuatan Model Machine Learning pada Tingkat Pertumbuhan Bayam Batik


```
from google.colab import drive
drive.mount('/content/drive')
```

```
!unzip /content/drive/MyDrive/camerapi.zip -d /content/camerapi
```

```
Archive:  /content/drive/MyDrive/camerapi.zip
  inflating: /content/camerapi/camerapi/2023-05-16_19-39-47
  inflating: /content/camerapi/camerapi/2023-05-16_20-39-48
  inflating: /content/camerapi/camerapi/2023-05-16_21-39-50
  inflating: /content/camerapi/camerapi/2023-05-16_22-39-51
  inflating: /content/camerapi/camerapi/2023-05-16_23-39-53
  inflating: /content/camerapi/camerapi/2023-05-17_00-39-55
  inflating: /content/camerapi/camerapi/2023-05-17_01-39-56
  inflating: /content/camerapi/camerapi/2023-05-17_02-39-58
  inflating: /content/camerapi/camerapi/2023-05-17_03-40-00
  inflating: /content/camerapi/camerapi/2023-05-17_04-40-01
  inflating: /content/camerapi/camerapi/2023-05-17_05-40-03
  inflating: /content/camerapi/camerapi/2023-05-17_06-40-05
  inflating: /content/camerapi/camerapi/2023-05-17_07-40-06
  inflating: /content/camerapi/camerapi/2023-05-17_08-40-08
  inflating: /content/camerapi/camerapi/2023-05-17_09-40-09
  inflating: /content/camerapi/camerapi/2023-05-17_10-40-11
  inflating: /content/camerapi/camerapi/2023-05-17_11-40-12
  inflating: /content/camerapi/camerapi/2023-05-17_12-40-14
  inflating: /content/camerapi/camerapi/2023-05-17_13-40-16
  inflating: /content/camerapi/camerapi/2023-05-17_14-40-17
  inflating: /content/camerapi/camerapi/2023-05-17_15-40-19
  inflating: /content/camerapi/camerapi/2023-05-17_16-40-20
  inflating: /content/camerapi/camerapi/2023-05-17_17-40-22
  inflating: /content/camerapi/camerapi/2023-05-17_18-40-24
  inflating: /content/camerapi/camerapi/2023-05-17_19-40-25
  inflating: /content/camerapi/camerapi/2023-05-17_20-19-17
  inflating: /content/camerapi/camerapi/2023-05-17_21-19-19
  inflating: /content/camerapi/camerapi/2023-05-17_22-19-20
  inflating: /content/camerapi/camerapi/2023-05-17_23-19-22
  inflating: /content/camerapi/camerapi/2023-05-18_00-19-23
  inflating: /content/camerapi/camerapi/2023-05-18_01-19-25
  inflating: /content/camerapi/camerapi/2023-05-18_02-19-27
  inflating: /content/camerapi/camerapi/2023-05-18_03-19-28
  inflating: /content/camerapi/camerapi/2023-05-18_04-19-30
  inflating: /content/camerapi/camerapi/2023-05-18_05-19-31
  inflating: /content/camerapi/camerapi/2023-05-18_06-19-33
  inflating: /content/camerapi/camerapi/2023-05-18_07-19-34
  inflating: /content/camerapi/camerapi/2023-05-18_08-19-36
  inflating: /content/camerapi/camerapi/2023-05-18_09-19-38
  inflating: /content/camerapi/camerapi/2023-05-18_10-19-39
  inflating: /content/camerapi/camerapi/2023-05-18_11-19-41
  inflating: /content/camerapi/camerapi/2023-05-18_12-19-42
  inflating: /content/camerapi/camerapi/2023-05-18_13-19-44
  inflating: /content/camerapi/camerapi/2023-05-18_14-19-46
  inflating: /content/camerapi/camerapi/2023-05-18_15-19-47
  inflating: /content/camerapi/camerapi/2023-05-18_16-19-49
  inflating: /content/camerapi/camerapi/2023-05-18_17-19-50
  inflating: /content/camerapi/camerapi/2023-05-18_18-19-52
  inflating: /content/camerapi/camerapi/2023-05-18_19-19-53
  inflating: /content/camerapi/camerapi/2023-05-18_20-19-55
  inflating: /content/camerapi/camerapi/2023-05-18_21-19-57
  inflating: /content/camerapi/camerapi/2023-05-18_22-19-58
  inflating: /content/camerapi/camerapi/2023-05-18_23-20-00
  inflating: /content/camerapi/camerapi/2023-05-19_00-20-01
  inflating: /content/camerapi/camerapi/2023-05-19_01-20-03
  inflating: /content/camerapi/camerapi/2023-05-19_02-20-05
  inflating: /content/camerapi/camerapi/2023-05-19_03-20-06
```

```
!rm /content/camerapi/camerapi/2023-05-28_20-07-24
```

```
import os

def makedir(dir):
  if not os.path.exists(dir):
    os.makedirs(dir, exist_ok=True)

makedir("/content/week2")
makedir("/content/week3")
makedir("/content/week4")
```

```
import re
import os

week_2_data = []
week_3_data = []
week_4_data = []

for data in os.listdir("/content/camerapi/camerapi"):
  re_data = re.search(r"(\d+)-(\d+)-(\d+)_(\d+)-(\d+)-(\d+)", data)
  month, date = int(re_data[2]), int(re_data[3])
  # print(month,date)

  if month == 5:
    if date < 20:
      week_2_data.append(data)
    elif date < 24:
      week_3_data.append(data)
    else:
      week_4_data.append(data)
  else:
    if date < 5:
      week_4_data.append(data)


import shutil

for data in range(len(week_2_data)):
  shutil.copy("/content/camerapi/camerapi/"+week_2_data[data], "/content/week2/"+week_2_data[data])

for data in range(len(week_3_data)):
  shutil.copy("/content/camerapi/camerapi/"+week_3_data[data], "/content/week3/"+week_3_data[data])

for data in range(len(week_4_data)):
  shutil.copy("/content/camerapi/camerapi/"+week_4_data[data], "/content/week4/"+week_4_data[data])


print(len(week_2_data))
```

```
77
```

```
print(len(week_3_data))
```

```
96
```

```
print(len(week_4_data))
```

```
173
```

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def segment_image(image_array):
    # Tentukan rentang warna yang ingin disegmentasi dalam format RGB
    color_ranges = [
        # Rentang warna minggu 2
        (np.array([65, 101, 50]), np.array([121, 144, 119])),
        (np.array([62, 88, 43]), np.array([124, 167, 99])),
        (np.array([68, 76, 75]), np.array([128, 146, 136])),
        (np.array([87, 97, 80]), np.array([133, 175, 139])),
        (np.array([70, 99, 71]), np.array([131, 183, 133])),
        (np.array([67, 80, 68]), np.array([99, 117, 101])),
        (np.array([84, 96, 88]), np.array([110, 122, 111])),
        (np.array([59, 70, 57]), np.array([143, 177, 130])),
        (np.array([72, 90, 66]), np.array([134, 164, 131])),
        (np.array([69, 127, 107]), np.array([139, 137, 140])),
        (np.array([61, 84, 48]), np.array([143, 188, 133])),
        (np.array([76, 88, 74]), np.array([119, 148, 123])),
        (np.array([85, 104, 78]), np.array([157, 171, 136])),
        (np.array([83, 103, 84]), np.array([151, 198, 151])),
        # Rentang warna minggu 3
        (np.array([97, 94, 64]), np.array([170, 196, 141])),
        # Rentang warna minggu 4
        (np.array([44, 137, 65]), np.array([96, 208, 168])),
        (np.array([23, 77, 60]), np.array([57, 114, 129]))
    ]
```

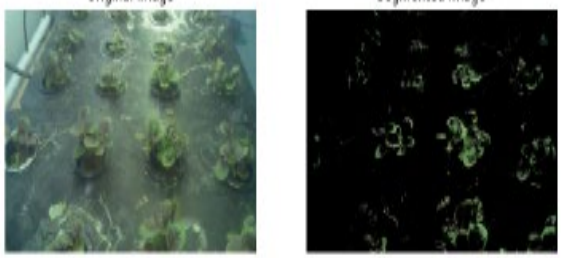
```
    # Rentang warna yang ingin dihindari dalam format RGB
    avoid_ranges = [
        # # Rentang warna minggu 2
        (np.array([52, 62, 57]), np.array([109, 107, 111])),
        (np.array([101, 120, 101]), np.array([126, 153, 127])),
        (np.array([76, 98, 95]), np.array([103, 123, 117])),
        (np.array([88, 127, 107]), np.array([93, 137, 140])),
        (np.array([63, 84, 89]), np.array([102, 136, 137])),
        (np.array([96, 150, 139]), np.array([116, 165, 170])),
        (np.array([113, 138, 121]), np.array([168, 198, 153])),
        (np.array([95, 116, 94]), np.array([117, 161, 161])),
        # Rentang warna minggu 3
        (np.array([75, 106, 96]), np.array([139, 142, 156])),
        (np.array([103, 162, 160]), np.array([112, 165, 164])),
        # Rentang warna minggu 4
        (np.array([15, 54, 61]), np.array([59, 115, 110]))
    ]

    # Membuat masker warna untuk rentang warna yang ingin dihindari
    avoid_color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in avoid_ranges:
        avoid_color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Membuat masker warna untuk rentang warna yang ingin disegmentasi
    color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in color_ranges:
        color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Menghilangkan area yang ingin dihindari dari masker warna hasil segmentasi
    color_mask &= ~avoid_color_mask

    return color_mask


def show_image(image_array,color_mask):
    # Buat gambar hasil segmentasi
    segmented_array = np.zeros_like(image_array)
    segmented_array[color_mask] = image_array[color_mask]

    # Tampilkan gambar asli dan gambar yang telah disegmentasi
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image_array)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(segmented_array)
    axes[1].set_title('Segmented Image')
    axes[1].axis('off')

    plt.show()


def upload_image(filename):
    try:
        image = Image.open(filename)
        image_array = np.array(image)
        color_mask = segment_image(image_array)
        return image_array, color_mask
    except Exception as e:
        print(e)
        return None, None


test = [
    "/content/camerapi/camerapi/2023-05-16_19-39-47",
    "/content/camerapi/camerapi/2023-05-22_00-21-56",
    "/content/camerapi/camerapi/2023-05-20_17-21-07",
    "/content/camerapi/camerapi/2023-05-28_21-07-26",
    "/content/camerapi/camerapi/2023-05-29_01-07-32",
    "/content/camerapi/camerapi/2023-06-04_02-11-25",
    "/content/camerapi/camerapi/2023-06-10_22-29-51",
    "/content/week2/2023-05-17_16-40-20",
    "/content/week2/2023-05-19_17-20-28",
    "/content/week3/2023-05-23_22-23-10"
]
```



```
for loop in test:
  image_array, color_mask = upload_image(loop)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(loop)
  show_image(image_array, color_mask)
```

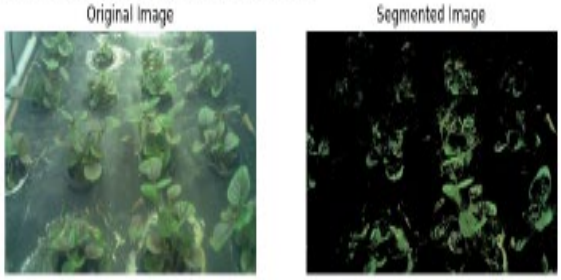
111683
/content/camerapi/camerapi/2023-05-16_19-39-47

Original Image Segmented Image



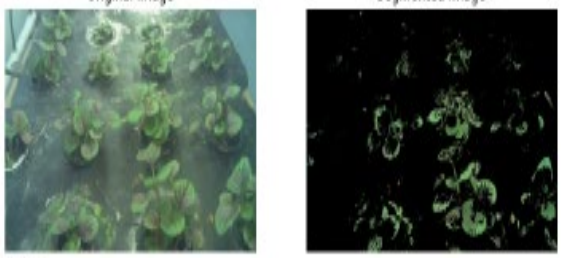
227937
/content/camerapi/camerapi/2023-05-22_00-21-56

Original Image Segmented Image



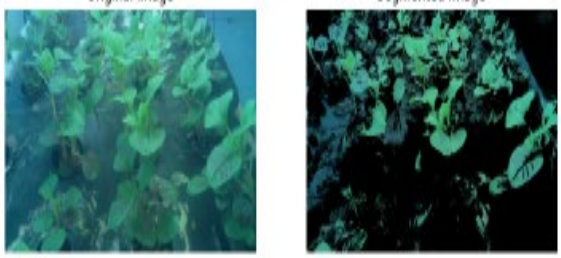
178569
/content/camerapi/camerapi/2023-05-20_17-21-07

Original Image Segmented Image



634827
/content/camerapi/camerapi/2023-05-28_21-07-26

Original Image Segmented Image



576540
/content/camerapi/camerapi/2023-05-29_01-07-32

Original Image Segmented Image

```
#evaluate the algorithm to classify

def classify_image(filename):
    _, color_mask = upload_image(filename)
    if color_mask is None:
      return "wrong data"
    num_pixels = np.sum(color_mask)

    if num_pixels <= 200000:
      return (0, num_pixels, filename)
    elif num_pixels <= 500000:
      return (1, num_pixels, filename)
    else:
      return (2, num_pixels, filename)


image_pixels = []
image_groups = []
predict_groups = []
wrong_image = []

import os

for image in os.listdir("/content/week2"):
    results = classify_image("/content/week2/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(0)
    predict_groups.append(results[0])

    if results[0] != 0:
      wrong_image.append([image, results[2]])


for image in os.listdir("/content/week3"):
    results = classify_image("/content/week3/"+image)
    if results == "wrong data":
      continue

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(1)
    predict_groups.append(results[0])

    if results[0] != 1:
      wrong_image.append([image, results[2]])

for image in os.listdir("/content/week4"):
    results = classify_image("/content/week4/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(2)
    predict_groups.append(results[0])

    if results[0] != 2:
      wrong_image.append([image, results[2]])
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-23-36ea22aaba6e> in <cell line: 8>()
      7
      8 for image in os.listdir("/content/week2"):
----> 9     results = classify_image("/content/week2/"+image)
     10
     11     # untuk clustering, 1 kali saja hehe

                                3 frames
/usr/local/lib/python3.10/dist-packages/numpy/core/overrides.py in all(*args, **kwargs)

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
# Import necessary libraries
import numpy as np
```

Lampiran 3. Python Notebook untuk Pembuatan Model Machine Learning pada Tingkat Pertumbuhan Bayam Merah


```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!unzip "/content/drive/MyDrive/bayam_merah_fix_data (1).zip" -d /content/bayam_merah
```

```
Archive:  /content/drive/MyDrive/bayam_merah_fix_data (1).zip
  extracting: /content/bayam_merah/content/week2/2023-11-13_17:44:05_1194.88_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_23:36:44_1185.45_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_00:32:43_1171.09_6.7_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_02:37:19_1187.34_6.3_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_16:43:53_1188.4_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_00:41:46_1160.28_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_00:50:15_1159.36_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_17:30:08_1170.2_6.42_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_14:43:31_1173.58_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_09:38:40_1169.35_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_06:46:28_1150.06_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_07:43:05_1152.95_6.05_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_23:50:03_1180.11_6.54_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_20:32:04_1168.02_6.69_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_19:44:27_1157.36_6.02_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_23:45:23_1181.55_6.64_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_03:37:31_1157.78_6.02_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_09:34:20_1189.66_6.17_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_20:40:53_1177.13_6.8_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_10:07:18_1183.41_6.68_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_14:35:28_1156.7_6.67_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_11:47:10_1156.44_6.6_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_04:46:05_1185.7_6.99_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_21:32:16_1184.69_6.96_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_10:38:52_1193.15_6.56_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_14:48:16_1152.78_6.16_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_16:29:54_1157.46_6.97_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_13:35:25_1164.59_6.79_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_11:43:52_1169.04_6.13_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_05:33:33_1193.63_6.71_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_17:48:55_1181.54_6.58_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_07:33:57_1191.94_6.41_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_10:34:50_1177.39_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_14:18:30_1158.12_6.62_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_11:52:19_1185.4_6.12_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_17:40:25_1186.81_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_19:40:41_1180.61_6.38_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_10:46:58_1160.09_6.04_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_23:41:35_1162.71_6.64_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_05:37:55_1157.74_6.9_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_19:31:53_1184.95_6.6_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_12:35:13_1188.21_6.55_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_12:44:03_1168.37_6.81_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_12:52:34_1157.94_6.06_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_06:42:54_1173.11_6.86_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_09:07:05_1159.83_6.89_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_07:46:31_1165.28_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_13:52:46_1191.02_6.81_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_18:31:40_1165.39_6.94_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_22:36:30_1178.26_6.57_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_04:50:56_1163.82_6.08_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_08:38:29_1188.26_6.05_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_19:49:17_1162.09_6.25_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_18:36:09_1189.5_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_13:47:42_1190.02_6.05_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_06:33:45_1164.99_6.63_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_08:46:34_1173.27_6.69_24.8_79.8.jpg
```

```
import os
```

```
len(os.listdir("/content/bayam_merah/content/week2"))
```

```
130
```

```
len(os.listdir("/content/bayam_merah/content/week3"))
```

```
123
```

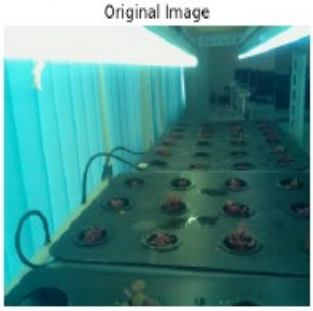
```
len(os.listdir("/content/bayam_merah/content/week4"))
```

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def segment_image(image_array):
    # Tentukan rentang warna yang ingin disegmentasi dalam format RGB
    color_ranges = [
        # Rentang warna minggu 2
        (np.array([70, 117, 17]), np.array([136, 213, 91])),
        (np.array([25, 96, 6]), np.array([72, 152, 63])),
        (np.array([58, 53, 13]), np.array([93, 122, 84])),
        (np.array([72, 90, 74]), np.array([89, 112, 87])),
        (np.array([83, 85, 27]), np.array([97, 117, 91])),
        (np.array([48, 115, 62]), np.array([66, 136, 77])),
        (np.array([77, 100, 85]), np.array([81, 108, 91])),
        # Rentang warna minggu 3
        (np.array([60, 114, 0]), np.array([102, 199, 51])),
        (np.array([109, 127, 74]), np.array([129, 158, 93])),
        (np.array([87, 146, 49]), np.array([108, 191, 85])),
        (np.array([73, 47, 67]), np.array([87, 136, 92])),
        (np.array([39, 54, 59]), np.array([63, 83, 81])),
        # Rentang warna minggu 4
        (np.array([104, 183, 77]), np.array([155, 238, 126])),
        (np.array([117, 188, 0]), np.array([176, 243, 7])),
        (np.array([195, 236, 145]), np.array([241, 245, 229])),
        (np.array([92, 189, 0]), np.array([120, 216, 55])),
        (np.array([24, 52, 6]), np.array([65, 103, 45])),
        (np.array([35, 78, 59]), np.array([55, 99, 81])),
        (np.array([43, 113, 17]), np.array([71, 153, 45])),
        (np.array([24, 52, 6]), np.array([65, 103, 45])),
        (np.array([129, 213, 0]), np.array([174, 243, 36])),
        (np.array([25, 60, 49]), np.array([52, 98, 72])),
        (np.array([55, 84, 52]), np.array([69, 101, 63])),
        (np.array([34, 87, 58]), np.array([53, 128, 70])),
        (np.array([62, 107, 61]), np.array([83, 154, 78])),
        (np.array([68, 96, 71]), np.array([83, 106, 81])),
        (np.array([26, 53, 44]), np.array([69, 101, 72])),
        (np.array([25, 71, 53]), np.array([65, 96, 75])),
        (np.array([50, 135, 59]), np.array([67, 144, 76])),
        (np.array([30, 51, 35]), np.array([69, 130, 80]))
    ]

     # Rentang warna yang ingin dihindari dalam format RGB
    avoid_ranges = [
        # Rentang warna minggu 2
        (np.array([58, 82, 58]), np.array([66, 88, 64])),
        # (np.array([41, 88, 73]), np.array([72, 108, 84])),
        (np.array([32, 69, 62]), np.array([59, 89, 89])),
        (np.array([58, 82, 58]), np.array([66, 88, 64])),
        (np.array([103, 136, 89]), np.array([158, 197, 127])),
        (np.array([44, 91, 77]), np.array([63, 102, 84])),
        # Rentang warna minggu 3
        (np.array([62, 96, 69]), np.array([74, 110, 84])),
        (np.array([68, 106, 77]), np.array([104, 141, 99])),
        (np.array([25, 79, 58]), np.array([52, 108, 72])),
        (np.array([17, 56, 53]), np.array([28, 71, 61])),
        (np.array([19, 63, 37]), np.array([30, 73, 63])),
        (np.array([21, 66, 57]), np.array([31, 80, 73])),
        (np.array([66, 93, 59]), np.array([109, 135, 87])),
        (np.array([19, 58, 56]), np.array([29, 68, 65])),
        (np.array([52, 87, 72]), np.array([63, 102, 87])),
        (np.array([20, 91, 76]), np.array([51, 105, 85])),
        # Rentang warna minggu 4
        (np.array([30, 65, 52]), np.array([40, 74, 62])),
        (np.array([33, 65, 44]), np.array([51, 93, 71])),
        (np.array([55, 90, 66]), np.array([62, 97, 73])),
        (np.array([31, 87, 76]), np.array([42, 92, 83])),
        (np.array([24, 54, 40]), np.array([36, 67, 53])),
        (np.array([22, 59, 54]), np.array([29, 67, 62])),
        (np.array([88, 119, 71]), np.array([166, 200, 114])),
    ]

    # Membuat masker warna untuk rentang warna yang ingin dihindari
    avoid_color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in avoid_ranges:
        avoid_color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)
```

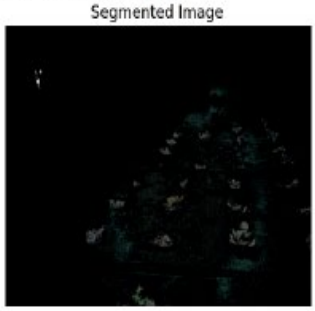
```
    # Membuat masker warna untuk rentang warna yang ingin disegmentasi
    color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in color_ranges:
        color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Menghilangkan area yang ingin dihindari dari masker warna hasil segmentasi
    color_mask &= ~avoid_color_mask

    return color_mask


def show_image(image_array,color_mask):
    # Buat gambar hasil segmentasi
    segmented_array = np.zeros_like(image_array)
    segmented_array[color_mask] = image_array[color_mask]

    # Tampilkan gambar asli dan gambar yang telah disegmentasi
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image_array)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(segmented_array)
    axes[1].set_title('Segmented Image')
    axes[1].axis('off')

    plt.show()


def upload_image(filename):
    try:
        image = Image.open(filename)
        image_array = np.array(image)
        color_mask = segment_image(image_array)
        return image_array, color_mask
    except Exception as e:
        print(e)
        return None, None


test = [
    "/content/bayam_merah/content/week2/2023-11-13_14:43:31_1173.58_6.66_24.8_79.8.jpg",
    "/content/bayam_merah/content/week2/2023-11-19_01:41:57_1173.21_6.72_24.8_79.8.jpg",
    "/content/bayam_merah/content/week3/2023-11-20_12:20:58_1169.13_6.38_24.8_79.8.jpg",
    "/content/bayam_merah/content/week3/2023-11-26_00:51:13_1188.81_6.94_24.8_79.8.jpg",
    "/content/bayam_merah/content/week4/2023-11-28_02:44:03_1184.31_6.69_24.8_79.8.jpg",
    "/content/bayam_merah/content/week4/2023-12-06_22:53:49_1170.07_6.65_24.8_79.8.jpg"
]

for loop in test:
  image_array, color_mask = upload_image(loop)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(loop)
  show_image(image_array, color_mask)
```

```
46704
/content/bayam_merah/content/week2/2023-11-13_14:43:31_1173.58_6.66_24.8_79.8.jpg
```

Original Image

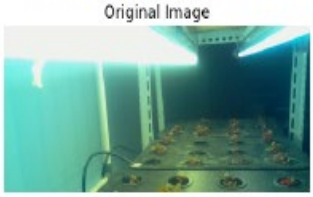


Segmented Image

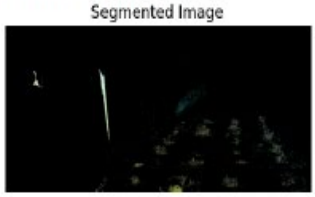


```
53488
/content/bayam_merah/content/week2/2023-11-19_01:41:57_1173.21_6.72_24.8_79.8.jpg
```

Original Image



Segmented Image



```
#evaluate the algorithm to classify

def classify_image(filename):
    _, color_mask = upload_image(filename)
    if color_mask is None:
      return "wrong data"
    num_pixels = np.sum(color_mask)

    if num_pixels <= 54000:
      return (0, num_pixels, filename)
    elif num_pixels <= 120000:
      return (1, num_pixels, filename)
    else:
      return (2, num_pixels, filename)
```

```
image_pixels = []
image_groups = []
predict_groups = []
wrong_image = []

import os

for image in os.listdir("/content/bayam_merah/content/week2"):
    results = classify_image("/content/bayam_merah/content/week2/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(0)
    predict_groups.append(results[0])
```

```
# Import necessary libraries
import numpy as np
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt


print(confusion_matrix(image_groups, predict_groups))
print(f"Accuracy (might need rearrangement): {accuracy_score(image_groups, predict_groups)}")
```

```
[[ 72  58   0]
 [ 21  98   4]
 [  0  47 164]]
Accuracy (might need rearrangement): 0.7198275862068966
```

```
    predict_groups.append(results[0])
```

```
# show the wrong data
for image, filename in wrong_image:
  image_array, color_mask = upload_image(filename)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(filename)
  show_image(image_array, color_mask)
```

```
image_pixel_values = np.array(image_pixels).reshape(-1, 1)
```

```
    predict_groups.append(results[0])
```

```
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Lampiran 4. Python Notebook untuk Pembuatan Model Machine Learning pada Tingkat Nutrisi Bayam


```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!unzip "/content/drive/MyDrive/bayam_merah_fix_data (1).zip" -d /content/bayam_merah
```

```
Archive:  /content/drive/MyDrive/bayam_merah_fix_data (1).zip
  extracting: /content/bayam_merah/content/week2/2023-11-13_17:44:05_1194.88_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_23:36:44_1185.45_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_00:32:43_1171.09_6.7_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_02:37:19_1187.34_6.3_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_16:43:53_1188.4_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_00:41:46_1160.28_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_00:50:15_1159.36_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_17:30:00_1170.2_6.42_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_14:43:31_1173.58_6.66_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_09:38:40_1169.35_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_06:46:28_1150.06_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_07:43:05_1152.95_6.05_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_23:50:03_1180.11_6.54_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_20:32:04_1168.02_6.69_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_19:44:27_1157.36_6.02_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-13_23:45:23_1181.55_6.64_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_03:37:31_1157.78_6.02_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_09:34:20_1189.66_6.17_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_20:40:53_1177.13_6.8_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_10:07:18_1183.41_6.68_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_14:35:28_1156.7_6.67_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_11:47:10_1156.44_6.6_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_04:46:05_1185.7_6.99_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_21:32:16_1184.69_6.96_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_10:38:52_1193.15_6.56_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_14:48:16_1152.78_6.16_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_16:29:54_1157.46_6.97_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_13:35:25_1164.59_6.79_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_11:43:52_1169.04_6.13_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_05:33:33_1193.63_6.71_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_17:48:55_1181.54_6.58_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_07:33:57_1191.94_6.41_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_10:34:50_1177.39_6.15_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_14:18:30_1158.12_6.62_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_11:52:19_1185.4_6.12_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_17:40:25_1186.81_6.23_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_19:40:41_1180.61_6.38_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_10:46:58_1160.09_6.04_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_23:41:35_1162.71_6.64_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_05:37:55_1157.74_6.9_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_19:31:53_1184.95_6.6_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_12:35:13_1188.21_6.55_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_12:44:03_1168.37_6.81_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_12:52:34_1157.94_6.06_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-19_06:42:54_1173.11_6.86_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_09:07:05_1159.83_6.89_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_07:46:31_1165.28_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_13:52:46_1191.02_6.81_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-16_18:31:40_1165.39_6.94_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_22:36:30_1178.26_6.57_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-15_04:50:56_1163.82_6.08_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-18_08:38:29_1188.26_6.85_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_19:49:17_1162.09_6.25_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_18:36:09_1189.5_6.4_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_13:47:42_1190.02_6.05_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-17_06:33:45_1164.99_6.63_24.8_79.8.jpg
  extracting: /content/bayam_merah/content/week2/2023-11-14_08:46:34_1173.27_6.69_24.8_79.8.jpg
```

```
import os
print(len(os.listdir("/content/drive/MyDrive/new malnutrisi")))
print(len(os.listdir("/content/drive/MyDrive/new overnutrisi")))
```

```
43
47
```

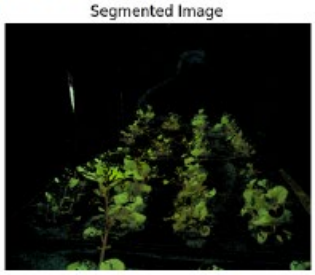
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def segment_image(image_array):
    # Tentukan rentang warna yang ingin disegmentasi dalam format RGB
    color_ranges = [
        # Rentang warna minggu 2
        (np.array([70, 117, 17]), np.array([136, 213, 91])),
        (np.array([25, 96, 6]), np.array([72, 152, 63])),
        (np.array([58, 53, 13]), np.array([93, 122, 84])),
        (np.array([72, 90, 74]), np.array([89, 112, 87])),
        (np.array([83, 85, 27]), np.array([97, 117, 91])),
        (np.array([48, 115, 62]), np.array([66, 136, 77])),
        (np.array([77, 100, 85]), np.array([81, 108, 91])),
        # Rentang warna minggu 3
        (np.array([60, 114, 0]), np.array([102, 199, 51])),
        (np.array([109, 127, 74]), np.array([129, 158, 93])),
        (np.array([87, 146, 49]), np.array([108, 191, 85])),
        (np.array([73, 47, 67]), np.array([87, 136, 92])),
        (np.array([39, 54, 59]), np.array([63, 83, 81])),
        # Rentang warna minggu 4
        (np.array([104, 183, 77]), np.array([155, 238, 126])),
        (np.array([117, 188, 0]), np.array([176, 243, 7])),
        (np.array([195, 236, 145]), np.array([241, 245, 229])),
        (np.array([92, 189, 0]), np.array([120, 216, 55])),
        (np.array([24, 52, 6]), np.array([65, 103, 45])),
        (np.array([35, 78, 59]), np.array([55, 99, 81])),
        (np.array([43, 113, 17]), np.array([71, 153, 45])),
        (np.array([24, 52, 6]), np.array([65, 103, 45])),
        (np.array([129, 213, 0]), np.array([174, 243, 36])),
        (np.array([25, 60, 49]), np.array([52, 98, 72])),
        (np.array([55, 84, 52]), np.array([69, 101, 63])),
        (np.array([34, 87, 58]), np.array([53, 128, 70])),
        (np.array([62, 107, 61]), np.array([83, 154, 78])),
        (np.array([68, 96, 71]), np.array([83, 106, 81])),
        (np.array([26, 53, 44]), np.array([69, 101, 72])),
        (np.array([25, 71, 53]), np.array([65, 96, 75])),
        (np.array([50, 135, 59]), np.array([67, 144, 76])),
        (np.array([30, 51, 35]), np.array([69, 130, 80]))
    ]

     # Rentang warna yang ingin dihindari dalam format RGB
    avoid_ranges = [
        # Rentang warna minggu 2
        (np.array([58, 82, 58]), np.array([66, 88, 64])),
        # (np.array([41, 88, 73]), np.array([72, 108, 84])),
        (np.array([32, 69, 62]), np.array([59, 89, 89])),
        (np.array([58, 82, 58]), np.array([66, 88, 64])),
        (np.array([103, 136, 89]), np.array([158, 197, 127])),
        (np.array([44, 91, 77]), np.array([63, 102, 84])),
        (np.array([196, 241, 207]), np.array([240, 244, 239])),
        (np.array([62, 88, 56]), np.array([75, 104, 75])),
        # Rentang warna minggu 3
        (np.array([62, 96, 69]), np.array([74, 110, 84])),
        (np.array([68, 106, 77]), np.array([104, 141, 99])),
        (np.array([25, 79, 58]), np.array([52, 108, 72])),
        (np.array([17, 56, 53]), np.array([28, 71, 61])),
        (np.array([19, 63, 37]), np.array([30, 73, 63])),
        (np.array([21, 66, 57]), np.array([31, 80, 73])),
        (np.array([66, 93, 59]), np.array([109, 135, 87])),
        (np.array([19, 58, 56]), np.array([29, 68, 65])),
        (np.array([52, 87, 72]), np.array([63, 102, 87])),
        (np.array([20, 91, 76]), np.array([51, 105, 85])),
        (np.array([45, 73, 46]), np.array([59, 84, 59])),
        (np.array([47, 65, 30]), np.array([60, 76, 35])),
        # Rentang warna minggu 4
        (np.array([30, 65, 52]), np.array([40, 74, 62])),
        (np.array([33, 65, 44]), np.array([51, 93, 71])),
        (np.array([55, 90, 66]), np.array([62, 97, 73])),
        (np.array([31, 87, 76]), np.array([42, 92, 83])),
        (np.array([24, 54, 40]), np.array([36, 67, 53])),
        (np.array([22, 59, 54]), np.array([29, 67, 62])),
        (np.array([88, 119, 71]), np.array([166, 200, 114])),
        (np.array([57, 72, 29]), np.array([63, 78, 33])),
    ]
```

```
    # Membuat masker warna untuk rentang warna yang ingin dihindari
    avoid_color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in avoid_ranges:
        avoid_color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Membuat masker warna untuk rentang warna yang ingin disegmentasi
    color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in color_ranges:
        color_mask |= np.all((image_array >= lower_color) & (image_array <= upper_color), axis=-1)

    # Menghilangkan area yang ingin dihindari dari masker warna hasil segmentasi
    color_mask &= ~avoid_color_mask

    return color_mask


def show_image(image_array,color_mask):
    # Buat gambar hasil segmentasi
    segmented_array = np.zeros_like(image_array)
    segmented_array[color_mask] = image_array[color_mask]

    # Tampilkan gambar asli dan gambar yang telah disegmentasi
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image_array)
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(segmented_array)
    axes[1].set_title('Segmented Image')
    axes[1].axis('off')

    plt.show()


def upload_image(filename):
    try:
        image = Image.open(filename)
        image_array = np.array(image)
        color_mask = segment_image(image_array)
        return image_array, color_mask
    except Exception as e:
        print(e)
        return None, None


test = [
    "/content/bayam_merah/content/week4/2023-12-06_22:53:49_1170.07_6.65_24.8_79.8.jpg",
    "/content/bayam_merah/content/week4/2023-12-07_09:56:05_1159.02_6.3_24.8_79.8.jpg",
    "/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20_07:38:27.jpg",
    "/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_08:45:24.jpg",
    "/content/drive/MyDrive/new overnutrisi/Copy of 2024-02-24_18:58:08.jpg",

]

for loop in test:
  image_array, color_mask = upload_image(loop)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(loop)
  show_image(image_array, color_mask)
```

```
178910
/content/bayam_merah/content/week4/2023-12-06_22:53:49_1170.07_6.65_24.8_79.8.jpg
```

Original Image

Segmented Image



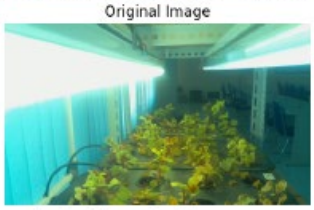
```
125016
/content/bayam_merah/content/week4/2023-12-07_09:56:05_1159.02_6.3_24.8_79.8.jpg
```

Original Image

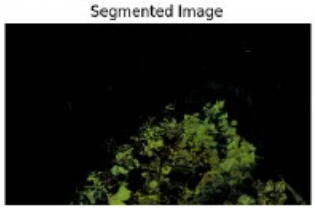
Segmented Image

```
182262
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20 07:38:27.jpg
```

Original Image



Segmented Image



```
#evaluate the algorithm to classify

def classify_image(filename):
    _, color_mask = upload_image(filename)
    if color_mask is None:
      return "wrong data"
    num_pixels = np.sum(color_mask)

    if num_pixels <= 180000:
      return (0, num_pixels, filename)
    elif num_pixels <= 240000:
      return (1, num_pixels, filename)
    else:
      return (2, num_pixels, filename)
```

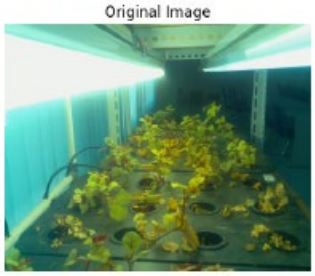
```
image_pixels = []
image_groups = []
predict_groups = []
wrong_image = []

import os

for image in os.listdir("/content/drive/MyDrive/new malnutrisi"):
    results = classify_image("/content/drive/MyDrive/new malnutrisi/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(1)
    predict_groups.append(results[0])

    if results[0] != 0:
      wrong_image.append([image, results[2]])

week4_files = os.listdir("/content/bayam_merah/content/week4")
# week4_files.sort(reverse=True)

for image in week4_files[:50]:
    results = classify_image("/content/bayam_merah/content/week4/"+image)
    if results == "wrong data":
      continue

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(0)
    predict_groups.append(results[0])

    if results[0] != 1:
      wrong_image.append([image, results[2]])

for image in os.listdir("/content/drive/MyDrive/new overnutrisi"):
    results = classify_image("/content/drive/MyDrive/new overnutrisi/"+image)

    # untuk clustering, 1 kali saja hehe
    image_pixels.append(results[1])
    image_groups.append(2)
    predict_groups.append(results[0])

    if results[0] != 2:
      wrong_image.append([image, results[2]])


# Import necessary libraries
import numpy as np
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt


print(confusion_matrix(image_groups, predict_groups))
print(f"Accuracy (might need rearrangement): {accuracy_score(image_groups, predict_groups)}")
```

```
[[47  3  0]
 [ 4 21 18]
 [ 1 12 34]]
Accuracy (might need rearrangement): 0.7285714285714285
```

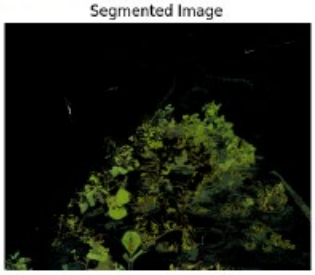
```
# show the wrong data
for image, filename in wrong_image:
  image_array, color_mask = upload_image(filename)
  num_pixels = np.sum(color_mask)
  print(num_pixels)
  print(filename)
  show_image(image_array, color_mask)
```

211160
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_06:44:56.jpg

Original Image

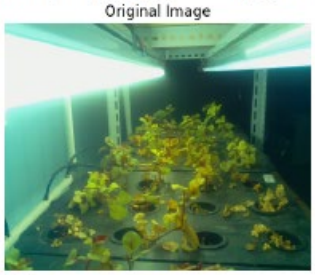


Segmented Image

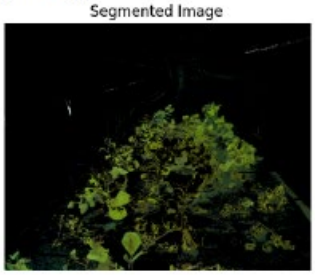


218427
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_05:44:43.jpg

Original Image

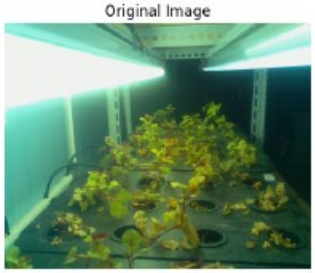


Segmented Image

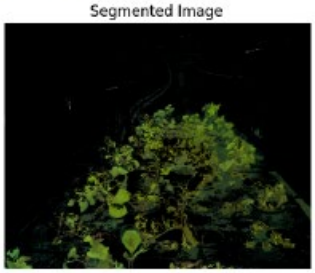


252989
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21 02:43:58.jpg

Original Image

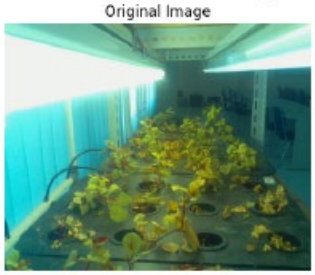


Segmented Image

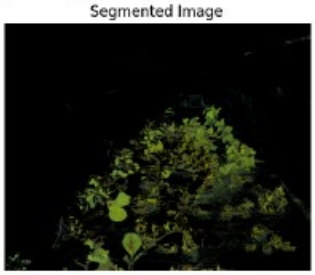


183803
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_08:45:24.jpg

Original Image

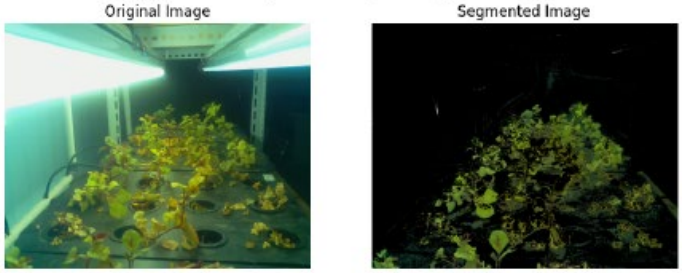


Segmented Image



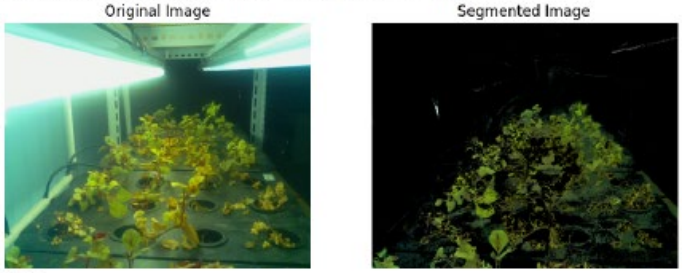
239934
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21 03:44:16.jpg

Original Image Segmented Image



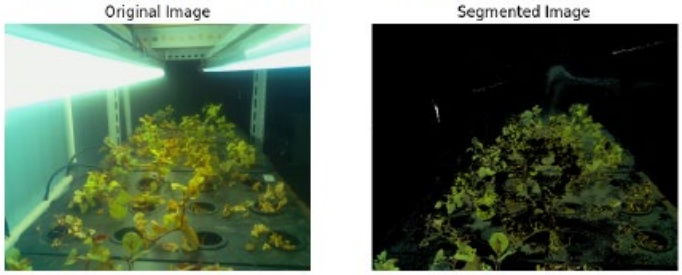
261384
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_04:44:30.jpg

Original Image Segmented Image



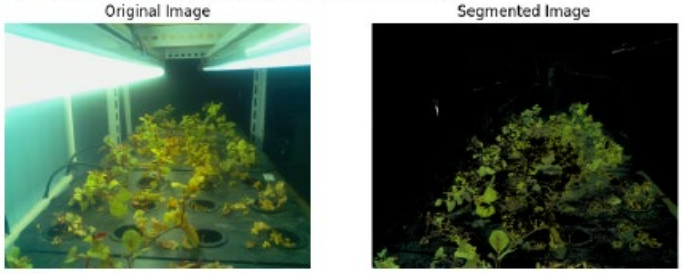
278388
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20_19:42:19.jpg

Original Image Segmented Image



245807
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20_18:42:06.jpg

Original Image Segmented Image

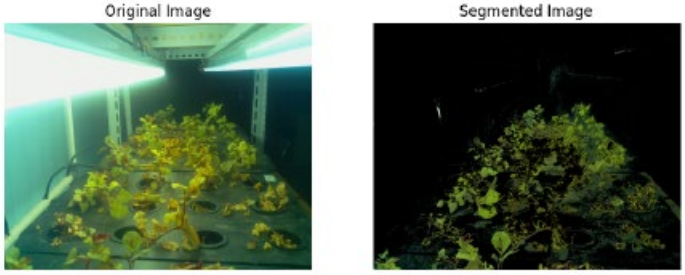


252218
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20 23:43:15.jpg

Original Image Segmented Image

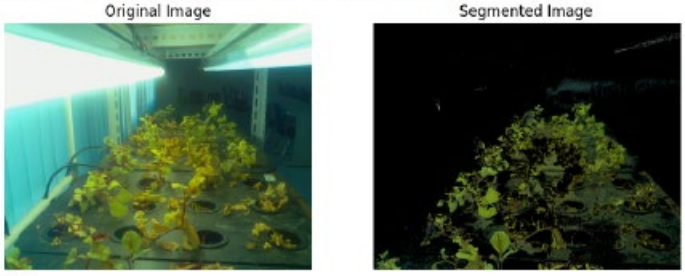
241028
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20 21:42:45.jpg

Original Image Segmented Segmented Image



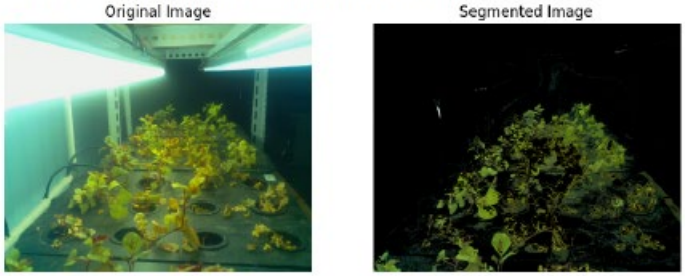
274196
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20_17:41:51.jpg

Original Image Segmented Image



239761
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-20_20:42:32.jpg

Original Image Segmented Image



250373
/content/drive/MyDrive/new malnutrisi/Copy of 2024-02-21_00:43:29.jpg

Original Image Segmented Image

Lampiran 5. Arduino Code untuk Pengambilan Data Sensor

```
#include <EEPROM.h>
#include "DFRobot_PH.h"
#include "DHT.h"
#include "DFRobot_EC.h"
#include "GravityTDS.h"

#define EcSensorPin A2
#define TdsSensorPin A1
#define PhSensorPin A0
#define DHTSensorPin 7
#define DHTTYPE DHT22

GravityTDS gravityTds;

int phval = 0;
unsigned long int avgval;
int buffer_arr[10],temp;

float voltage,ec_value,ph_value,temperature = 25;
DFRobot_EC ec;
DFRobot_PH ph;
DHT dht(DHTSensorPin, DHTTYPE);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.setTimeout(100);

  gravityTds.setPin(TdsSensorPin);
  gravityTds.setAref(5.0);
  gravityTds.setAdcRange(1024);
  gravityTds.begin();

  ec.begin();
  ph.begin();
  dht.begin();
}

void loop() {
  // put your main code here, to run repeatedly:
  float tds, ph, ec, temp, hum;
  String data;

  tds = tdsValue();
  ph = phValue();
  ec = ecValue();
  temp = dht.readTemperature();
  hum = dht.readHumidity();

  reportToRaspi(tds, ph, ec, temp, hum);
  delay(1000);
}

void reportToRaspi(float tds, float ph, float ec, float temp, float hum){
  Serial.print(tds);
  Serial.print("-");
  Serial.print(ph);
  Serial.print("-");
  Serial.print(ec);
  Serial.print("-");
  Serial.print(temp);
  Serial.print("-");
  Serial.print(hum);
  Serial.println("");
}

float tdsValue(){
```

```
68.   gravityTds.setTemperature(temperature);  // set the temperature and execute
temperature compensation
69.   gravityTds.update();  //sample and calculate
70.   float tdsValue = gravityTds.getTdsValue();  // then get the value
71.   return(tdsValue);
72. }
73.
74. float phValue(){
75.   static unsigned long timepoint = millis();
76.     if(millis()-timepoint>1000U){                  //time interval: 1s
77.         timepoint = millis();
78.         //temperature = readTemperature();         // read your temperature
sensor to execute temperature compensation
79.         voltage = analogRead(PhSensorPin)/1024.0*5000;  // read the voltage
80.         ph_value = ph.readPH(voltage,temperature);  // convert voltage to pH
with temperature compensation
81.     }
82.   ph.calibration(voltage,temperature);           // calibration process by
Serail CMD
83.   return(ph_value);
84. }
85.
86. float ecValue(){
87.   static unsigned long timepoint = millis();
88.   if(millis()-timepoint>1000U)  //time interval: 1s
89.   {
90.     timepoint = millis();
91.     voltage = analogRead(EcSensorPin)/1024.5*5000;   // read the voltage
92.     //temperature = readTemperature();          // read your temperature sensor
to execute temperature compensation
93.     ec_value =  ec.readEC(voltage,temperature);  // convert voltage to EC with
temperature compensation
94.
95.   }
96.   ec.calibration(voltage,temperature);
97.   return(ec_value);
98. }
99.
```

Lampiran 6. Kode Klasifikasi Tingkat Pertumbuhan Tanaman

```
import pickle

from picamera import PiCamera
from PIL import Image
import RPi.GPIO as GPIO
import numpy as np
import time
import requests
import json

def set_relay(condition, relay):
    if condition:
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(relay, GPIO.OUT)
    else:
        GPIO.cleanup()


def take_image():
    filename = "/home/pi/tmp/image.jpg"
    time.sleep(1)
    with PiCamera() as camera:
        camera.rotation = 180
        camera.capture(filename)
    image = Image.open(filename)
    image_array = np.array(image)
    return image_array


def segment_image(image_array):
    # Tentukan rentang warna yang ingin disegmentasi dalam format RGB
    color_ranges = [
        # Rentang warna minggu 2
        (np.array([16, 83, 1]), np.array([107, 133, 86])),
        (np.array([21, 61, 0]), np.array([64, 147, 56])),
        (np.array([112, 156, 0]), np.array([151, 205, 10])),
        (np.array([72, 90, 74]), np.array([89, 112, 87])),
        # Rentang warna minggu 3
        (np.array([28, 114, 29]), np.array([72, 182, 83])),
        (np.array([48, 193, 106]), np.array([80, 205, 161])),
        # Rentang warna minggu 4
        (np.array([92, 207, 47]), np.array([240, 247, 239])),
        (np.array([57, 145, 15]), np.array([157, 246, 116])),
        (np.array([12, 166, 102]), np.array([40, 226, 151])),
        (np.array([5, 113, 71]), np.array([34, 175, 126])),
        (np.array([0, 121, 73]), np.array([8, 144, 124])),
        (np.array([0, 110, 66]), np.array([14, 138, 69])),
        (np.array([1, 105, 58]), np.array([18, 118, 79])),
        (np.array([27, 237, 187]), np.array([142, 243, 233])),
        (np.array([0, 167, 132]), np.array([9, 193, 157])),
        (np.array([0, 87, 55]), np.array([3, 111, 71])),
        (np.array([0, 129, 108]), np.array([14, 199, 153]))    ]

     # Rentang warna yang ingin dihindari dalam format RGB
    avoid_ranges = [
        # Rentang warna minggu 2
        (np.array([3, 180, 186]), np.array([73, 215, 217])),
        (np.array([54, 112, 77]), np.array([70, 133, 99])),
        (np.array([57, 106, 81]), np.array([74, 125, 92])),
        (np.array([40, 92, 75]), np.array([74, 130, 101])),
        (np.array([23, 93, 79]), np.array([47, 104, 87])),
        (np.array([69, 112, 69]), np.array([82, 125, 81])),
        (np.array([30, 85, 80]), np.array([40, 90, 90])),
        (np.array([96, 128, 82]), np.array([116, 149, 93])),
        (np.array([62, 100, 76]), np.array([106, 138, 93])),
        (np.array([183, 234, 168]), np.array([191, 241, 186])),
        (np.array([110, 137, 79]), np.array([187, 212, 103])),
        (np.array([137, 154, 79]), np.array([197, 201, 130])),
```

```
        (np.array([103, 211, 197]), np.array([135, 241, 232])),
        (np.array([43, 87, 66]), np.array([78, 100, 78])),
        (np.array([18, 83, 79]), np.array([28, 90, 86])),
        (np.array([85, 87, 32]), np.array([95, 115, 66])),
        (np.array([20, 78, 71]), np.array([41, 93, 79])),
        (np.array([86, 129, 89]), np.array([115, 159, 101])),
        (np.array([121, 202, 158]), np.array([127, 214, 182])),
        (np.array([46, 83, 52]), np.array([111, 126, 65])),
        (np.array([31, 82, 64]), np.array([73, 115, 90])),
        (np.array([85, 87, 32]), np.array([95, 115, 66])),
        (np.array([20, 78, 71]), np.array([41, 93, 79])),
        (np.array([86, 129, 89]), np.array([115, 159, 101])),
        (np.array([121, 202, 158]), np.array([127, 214, 182])),
        (np.array([46, 83, 52]), np.array([111, 126, 65])),
        (np.array([31, 82, 64]), np.array([73, 115, 90])),
        (np.array([146, 222, 177]), np.array([198, 243, 233])),
        (np.array([70, 232, 231]), np.array([111, 245, 243])),
        (np.array([86, 124, 67]), np.array([117, 162, 93])),
        (np.array([9, 83, 72]), np.array([48, 106, 83])),
        # Rentang warna minggu 3
        (np.array([27, 66, 37]), np.array([40, 83, 53])),
        (np.array([20, 65, 45]), np.array([32, 81, 63])),
        (np.array([25, 67, 47]), np.array([52, 92, 65])),
        (np.array([240, 242, 239]), np.array([240, 242, 239]))

    ]

    # Membuat masker warna untuk rentang warna yang ingin dihindari
    avoid_color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in avoid_ranges:
        avoid_color_mask |= np.all((image_array >= lower_color) & (image_array
<= upper_color), axis=-1)
    # Membuat masker warna untuk rentang warna yang ingin disegmentasi
    color_mask = np.zeros(image_array.shape[:2], dtype=bool)
    for lower_color, upper_color in color_ranges:
        color_mask |= np.all((image_array >= lower_color) & (image_array <=
upper_color), axis=-1)
    # Menghilangkan area yang ingin dihindari dari masker warna hasil
segmentasi
    color_mask &= ~avoid_color_mask
    return color_mask
def classify_image(color_mask):
    pickled_model = pickle.load(open("/home/pi/mu_code/bayam_putih.pkl", "rb"))
    pickled_model.predict(color_mask)
    print(pickled_model)
while True:
    #take the image then segment it
    img_array = take_image()
    color_mask = segment_image(img_array)

    # after segment, classify it
    classify_image(color_mask)

time.sleep(2)
GPIO.cleanup()

```

Lampiran 7. Setup Aplikasi Monitoring Data Berbasis Desktop

App.py

```
1. from process.global_val import data_storage from process.threads import
setup_read_thread from utils.graph import all_graph_update, create_graph from
utils.label import create_label, all_label_update


class App:
    def __init__(self, root):
        root.title("Hydroponic Monitoring App")
        root.configure(bg="white")
        width = 800         height = 800         screenwidth =
root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
        alignstr = "%dx%d+%d+%d" % (width,
                                    height,
                                    (screenwidth - width) / 2,
                                    (screenheight - height) / 2)

        root.geometry(alignstr)
        root.resizable(width=True, height=True)

        root.read_thread = setup_read_thread(data_storage)

        label_1 = create_label(root, 0)
        ax_1, canvas_1 = create_graph(root, 1)

        label_2 = create_label(root, 2)
        ax_2, canvas_2 = create_graph(root, 3)

        label_3 = create_label(root, 4)
        ax_3, canvas_3 = create_graph(root, 5)

        label_4 = create_label(root, 6)
        ax_4, canvas_4 = create_graph(root, 7)

        labels = [label_1, label_2, label_3, label_4]
        texts = ["tds", "ph", "temp", "hum"]
        axes = [ax_1, ax_2, ax_3, ax_4]
        canvas_widgets = [canvas_1, canvas_2, canvas_3, canvas_4]

        all_label_update(root, labels, texts)
        all_graph_update(root, texts, axes, canvas_widgets)
2.
3.
```

graph.py

```
1. import matplotlib.pyplot as plt import numpy as np from
matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from database.api import get_data from process.global_val import data_storage


def plot_data(data, ax):
    """Create a visualization graph for the data"""     graph_data = data

    value = np.flipud(np.array(graph_data, dtype="float"))

    ax.plot(value)
    return ax


def all_graph_update(root, kinds, axes, canvas_widgets):
    """update the contents of the graph every 10 second"""     data =
get_data("get_all", data_storage)
    print(data)
    if data:
```

```
        for kind, ax, canvas in  zip(kinds, axes, canvas_widgets):
            if kind in data:
                ax.clear()
                plot_data(data[kind], ax)
                canvas.draw()
            else:
                print("No data kind: "+kind)
    else:
        print("No data acquired")

    root.after(10000, lambda: all_graph_update(
        root,          kinds,          axes,          canvas_widgets         )
    )


def create_graph(root, loc):
    """create a sample figure for graph"""     fig = plt.figure(figsize=(7.9, 1.6))
    ax = fig.add_subplot(111)
    canvas = FigureCanvasTkAgg(fig, root)
    canvas.get_tk_widget().grid(row=loc, column=0)
    canvas.draw()
    return ax, canvas
2.
3.
```

label.py

```
1. from tkinter import Label from tkinter.font import Font

from database.api import get_data from process.global_val import data_storage


def create_label(root, loc):
    """Create a new label to show the data"""     label = Label(root)
    ft = Font(family="Helvetica", size=20)
    label["font"] = ft     label["fg"] = "#333333"     label["justify"] = "center"
label["bg"] = "white"     label.grid(row=loc, column=0)
    return label


def all_label_update(root, labels, texts):
    """Update the label text"""     data = get_data("get_one", data_storage)
    if data:
        for label, text in zip(labels, texts):
            if text in data:
                label.configure(text=text.upper() + ": " + str(data[text]))
            else:
                print(f"No data for key: {text}")
    else:
        print("No data retrieved from queue.")

    root.after(1000, lambda: all_label_update(root, labels, texts))
2.
3.
```

api.py

```
1. import re


def process_data(data, data_storage):
    """processing the data achieved"""     data = data.decode().rstrip()
    data = re.match(r"(.*)-(.*)-(.*)-(.*)-(.*)", data)

    if data is not None:
        data = data.groups()

        data_json = {
            "tds": data[0],
            "ph": data[1],
            "temp": data[3],
```

```
            "hum": data[4]
        }

    data_storage.add_data_point(data_json)


def get_data(req, data_storage):
    """
    get the data from the api,
    get_one to get last input data,
    get_all to get last 10 data
    """     if req == "get_one":
        return data_storage.get_single_data()
    elif req == "get_all":
        return data_storage.get_all_data()
2.
3.
```

DataStorage.py

```
1. class DataStorage:
    def __init__(self):
        self.last_10_data_points = []

    def add_data_point(self, data_point):
        self.last_10_data_points.append(data_point)
        while len(self.last_10_data_points) > 10:
            self.last_10_data_points.pop(0)

    def get_all_data(self):
        try:
            result = {key: [d[key] for d in self.last_10_data_points] for key in
self.last_10_data_points[0]}
            return result         except IndexError as e:
            print(e)
            return {}

    def get_single_data(self):
        if self.last_10_data_points:
            return self.last_10_data_points[-1]
        return None  # or return a default value or raise an exception

    def get_data_by_index(self, index):
        if 0 <= index < len(self.last_10_data_points):
            return self.last_10_data_points[index]
        return None  # or return a default value or raise an exception
2.
3.
```

arduino_local.py

```
1. import socket from csv import writer from datetime import datetime

import paho.mqtt.publish as publish

from database.api import process_data from process.global_val import data_queue

MQTT_SERVER = "10.1.2.60" MQTT_PATH = "sensor"


def publish_single(payload):
    publish.single(
        topic=MQTT_PATH,
        payload=payload,
        hostname=MQTT_SERVER     )


def read_arduino(arduino_serial, data_storage):
    while True:
        data = arduino_serial.readline()
        if data:
```

```
            data_csv = [str(datetime.now()), data]

            try:
                publish_single(data)
            except socket.timeout as e:
                print("Socket operation timed out!")

            with open("tmp/data_hidroponik.csv", "a") as file:
                writer_obj = writer(file)
                writer_obj.writerow(data_csv)

            for _ in range(4):
                processed_data = process_data(data, data_storage)
                data_queue.put(processed_data)
2.
3.
```

global_val.py

```
1. from database.DataStorage import DataStorage

import queue import serial

data_queue = queue.Queue()
ser = serial.Serial("/dev/ttyUSB0", baudrate=115200)
data_storage = DataStorage()
2.
3.
```

threads.py

```
1. import threading

from process.arduino_local import read_arduino from process.global_val import ser


def setup_read_thread(data_storage):
    read_thread = threading.Thread(
        target=read_arduino,
        args=(ser, data_storage)
    )
    read_thread.daemon = True  # Close the thread when the application closes
    read_thread.start()
    return read_thread
2.
3.
```

main.py

```
1. from ui.App import App from tkinter import Tk

# Press the green button in the gutter to run the script.
root = Tk()
app = App(root)

if __name__ == '__main__':
    root.mainloop()

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
2.
3.
```

Lampiran 8. Bash Script untuk Pengambilan Data Resource Usage

```
#!/bin/bash

# Set the start time
start_time=$(date +%s)

while true; do
    current_time=$(date +%H:%M:%S)
    elapsed_time=$(( $(date +%s) - start_time ))

    # Check if 30 minutes have passed
    if [ "$elapsed_time" -ge 1800 ]; then
        break
    fi

    cpu_usage=$(top -b -n1 | awk '/^%Cpu/{print $2}')
    memory_usage=$(free | awk '/Mem/{printf "%.2f", $3/$2*100}')

    echo "$current_time CPU: $cpu_usage%  Memory: $memory_usage%"
    sleep 1
done

```

Lampiran 9. Python Script untuk Pengambilan Gambar Tanaman

```
from picamera import PiCamera
import time
import datetime
import time
import os
camera = PiCamera()
start_time = datetime.time(7, 0)  # 7:00 AM
end_time = datetime.time(18, 0)  # 6:00 PM
log_file = '/home/pi/camerapi/error_log.txt'
def job():
    print("condition1")
    now = datetime.datetime.now()
    camera.rotation = 180
    filename = '/home/pi/camerapi/{}.jpg'.format(now.strftime('%Y-%m-%d_%H:%M:%>
    # Wait for 1 second before capturing the image
    time.sleep(1)
    # Capture the image
    camera.capture(filename)
    # Wait for 1 second before checking the time again
    time.sleep(1)
    print("condition2")
    while True:
        try:
            cmd = "rclone copy /home/pi/camerapi FauzulIchwan:/Sampel_TA"
            os.system(cmd)
        except exception as e:
            continue
        break
    print("ok")
while True:
    job()
    time.sleep(3600)

```