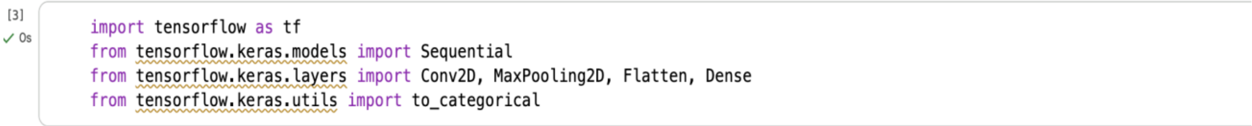
Edge Intelligence

Assessment (Lab Task - 2)

M Rahul – 25MML0011

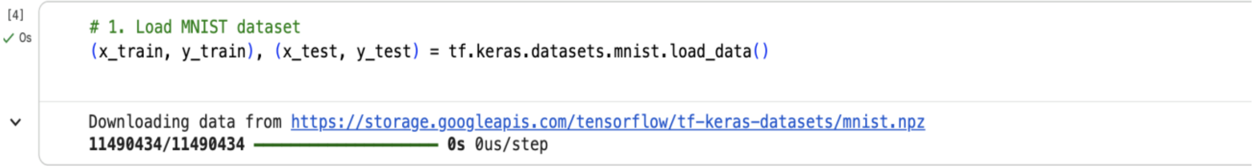
Task 1 :-

Analyzing an image dataset-Mnist , applying basic preprocessing operations, using either ANN and CNN. Saving the model using pickle model. Key takeaway: understanding the dataset and saving the model using pickle module.



[3] ✓ 0s

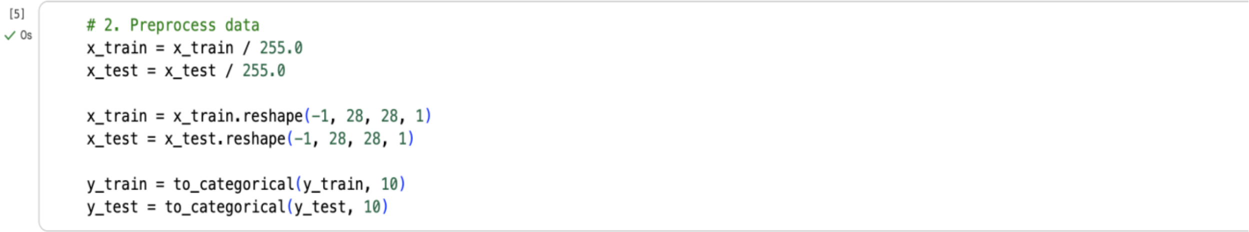
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
```



[4] ✓ 0s

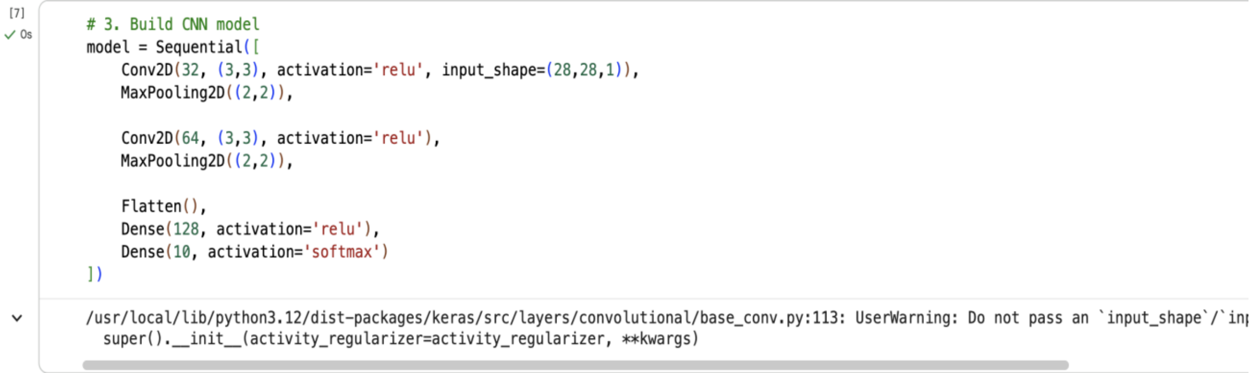
```
# 1. Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```



[5] ✓ 0s

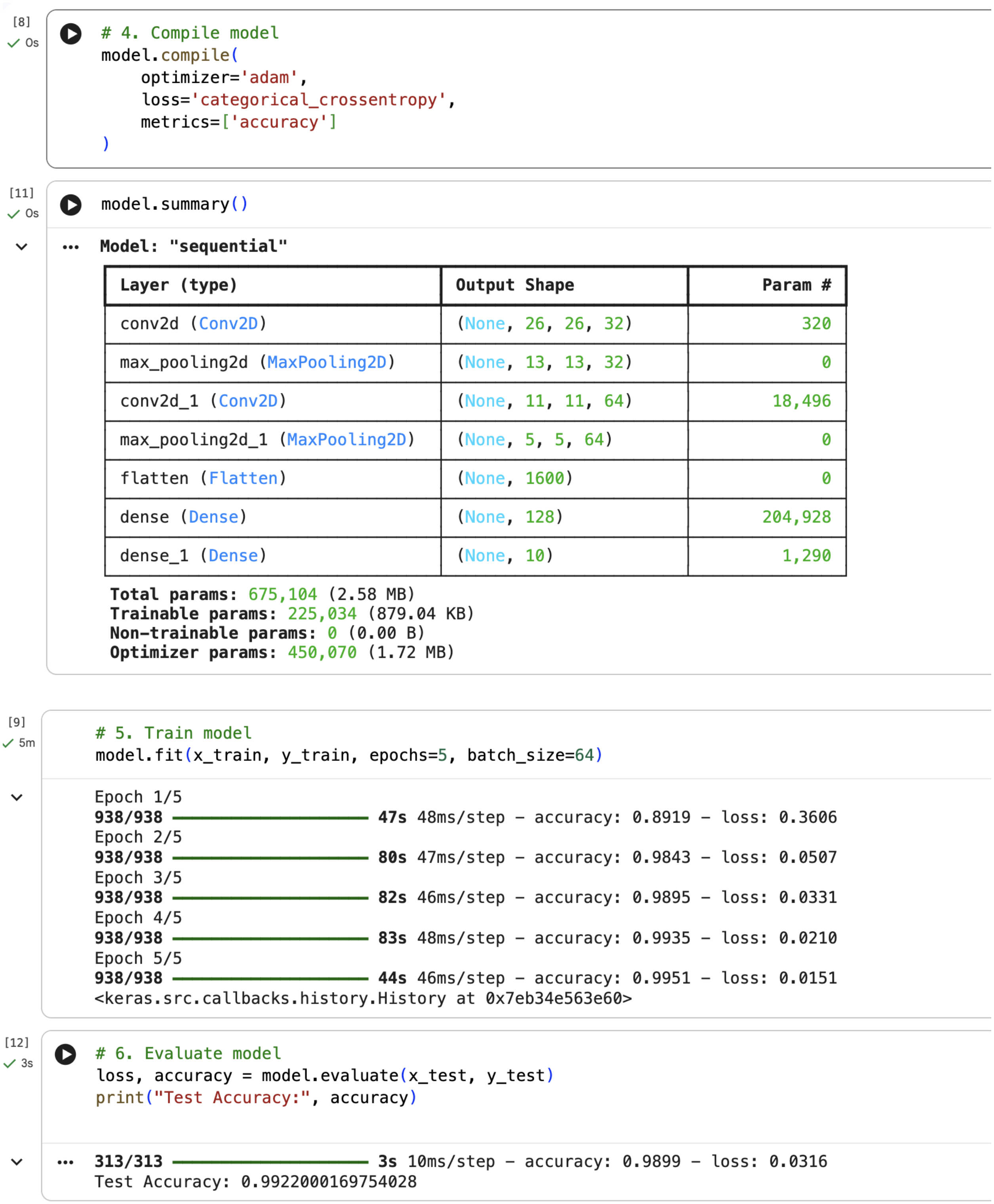
```
# 2. Preprocess data
x_train = x_train / 255.0
x_test = x_test / 255.0

x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```



[7] ✓ 0s

```
# 3. Build CNN model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D((2,2)),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`inp
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```



```
# 4. Compile model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
model.summary()
```

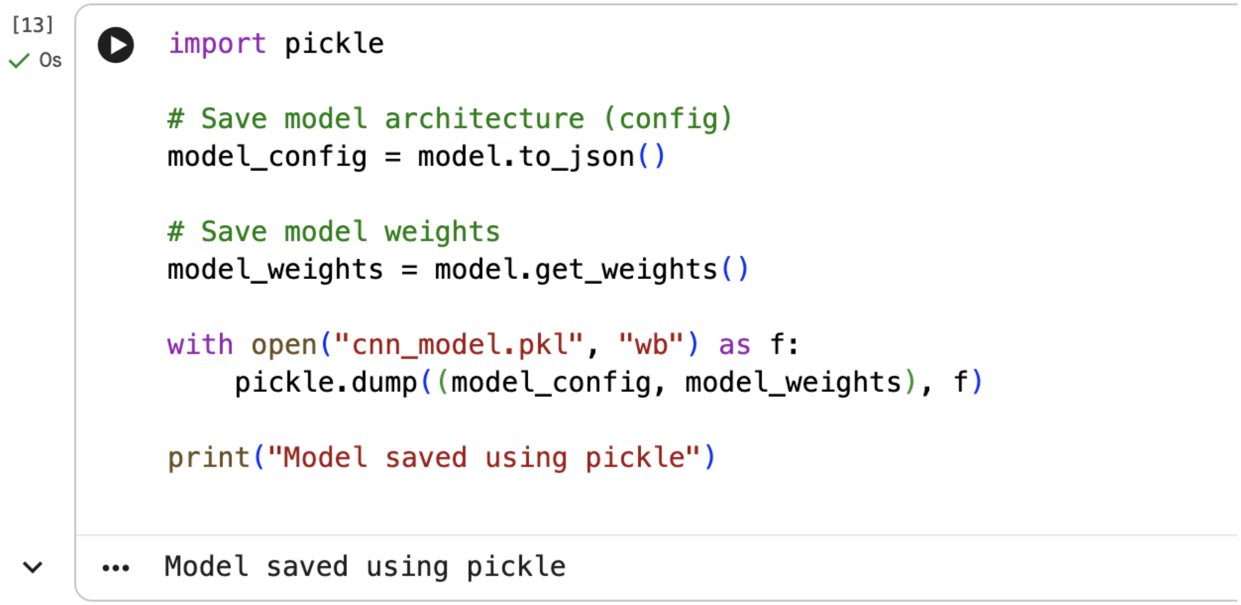
```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                  │ (None, 26, 26, 32)      │           320 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 13, 13, 32)      │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)                │ (None, 11, 11, 64)      │        18,496 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 5, 5, 64)        │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ flatten (Flatten)                │ (None, 1600)            │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 128)             │       204,928 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 10)              │         1,290 │
└──────────────────────────────────┴─────────────────────────┴───────────────┘
 Total params: 675,104 (2.58 MB)
 Trainable params: 225,034 (879.04 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 450,070 (1.72 MB)
```

```
# 5. Train model
model.fit(x_train, y_train, epochs=5, batch_size=64)
```

```
Epoch 1/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 47s 48ms/step - accuracy: 0.8919 - loss: 0.3606
Epoch 2/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 80s 47ms/step - accuracy: 0.9843 - loss: 0.0507
Epoch 3/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 82s 46ms/step - accuracy: 0.9895 - loss: 0.0331
Epoch 4/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 83s 48ms/step - accuracy: 0.9935 - loss: 0.0210
Epoch 5/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 44s 46ms/step - accuracy: 0.9951 - loss: 0.0151
<keras.src.callbacks.history.History at 0x7eb34e563e60>
```

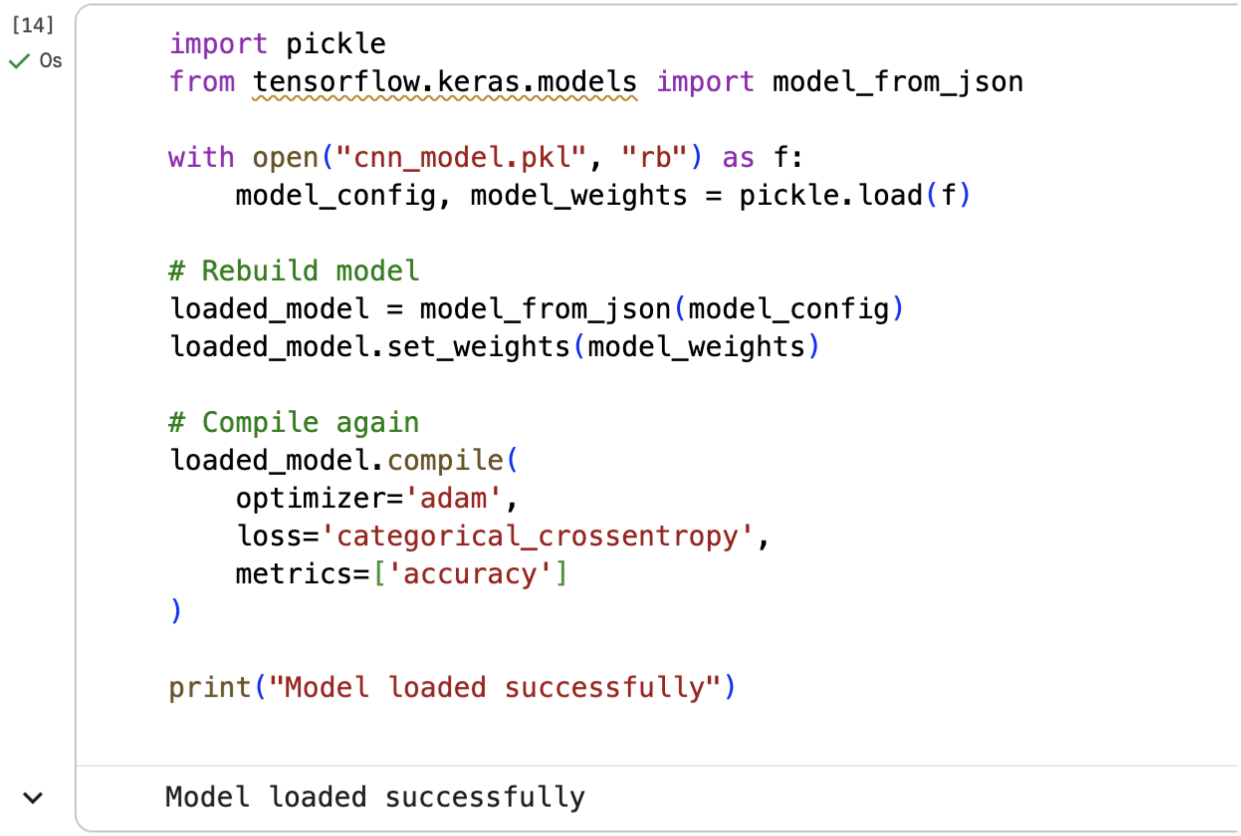
```
# 6. Evaluate model
loss, accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", accuracy)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - accuracy: 0.9899 - loss: 0.0316
Test Accuracy: 0.9922000169754028
```



```python
import pickle

# Save model architecture (config)
model_config = model.to_json()

# Save model weights
model_weights = model.get_weights()

with open("cnn_model.pkl", "wb") as f:
    pickle.dump((model_config, model_weights), f)

print("Model saved using pickle")
```

```
Model saved using pickle
```



```python
import pickle
from tensorflow.keras.models import model_from_json

with open("cnn_model.pkl", "rb") as f:
    model_config, model_weights = pickle.load(f)

# Rebuild model
loaded_model = model_from_json(model_config)
loaded_model.set_weights(model_weights)

# Compile again
loaded_model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

print("Model loaded successfully")
```

```
Model loaded successfully
```

Task 2 :-

1.Create account in edge impulse,

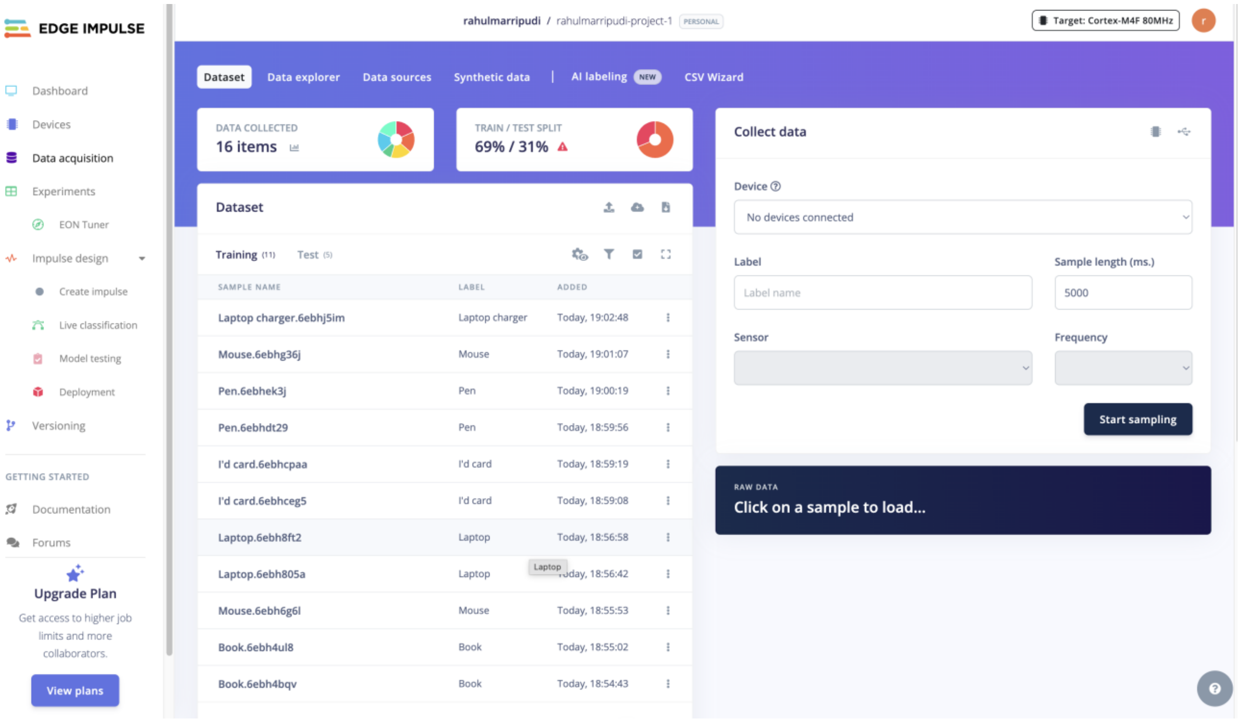
2.go to data acquisition .

3.Choose connect data option.

4.scan qr using phone.

5. Select Label data before clicking a photo.

6. Split the clicked photos into train and test data



rahulmarripudi / rahulmarripudi-project-1 PERSONAL

Target: Cortex-M4F 80MHz

DATA COLLECTED
16 items

TRAIN / TEST SPLIT
69% / 31%

Dataset

Training (11) Test (5)

SAMPLE NAME	LABEL	ADDED
Laptop charger.6ebhj5im	Laptop charger	Today, 19:02:48
Mouse.6ebhg36j	Mouse	Today, 19:01:07
Pen.6ebhek3j	Pen	Today, 19:00:19
Pen.6ebhdt29	Pen	Today, 18:59:56
I'd card.6ebhcpaa	I'd card	Today, 18:59:19
I'd card.6ebhceg5	I'd card	Today, 18:59:08
Laptop.6ebh8ft2	Laptop	Today, 18:56:58
Laptop.6ebh805a	Laptop	Today, 18:56:42
Mouse.6ebh6g6l	Mouse	Today, 18:55:53
Book.6ebh4ul8	Book	Today, 18:55:02
Book.6ebh4bqv	Book	Today, 18:54:43

Collect data

Device: No devices connected

Label: Label name

Sample length (ms.): 5000

Sensor

Frequency

RAW DATA
Click on a sample to load...