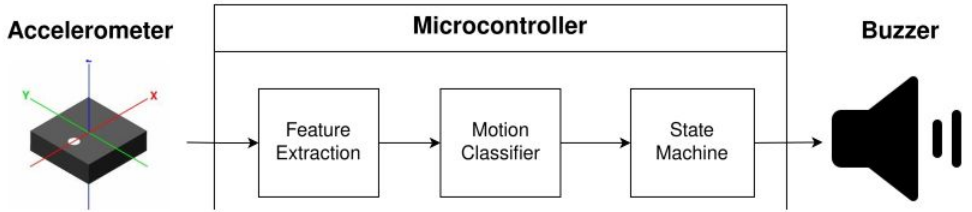
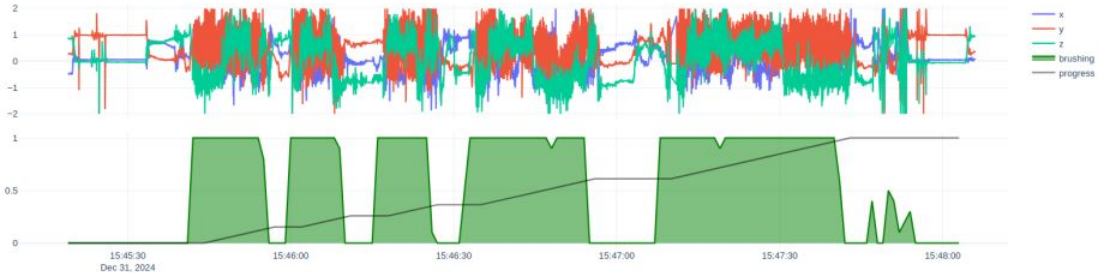
Automatic toothbrushing timer

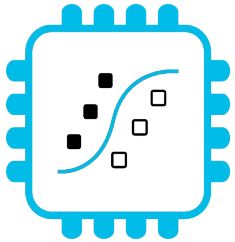
Using accelerometer and machine learning

https://github.com/jonnor/toothbrush





Powered by emlearn



emlearn -micropython

emlearn_iir	Infinite Impulse Response filters
emlearn_trees	Random Forest
emlearn_fft	Fast Fourier Transform
emlearn_cnn	Convolutional Neural Networks
emlearn_neighbors	K-nearest Neighbors

1. Train on PC

$ pip install emlearn

Simple training

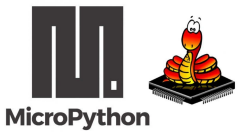
- Model creation **in Python**
- **Use standard libraries**
  a. scikit-learn
  b. Keras
- One-line to **export to device**

2. Deploy on device

$ mip install https://..../emlearn_trees.mpy

Convenient & Efficient

- **MicroPython API**
- **Single .mpy file install**
- **Fast**. Implemented in C
- **Small**. 2 kB+ FLASH



Activity tracker

Accelerometer

Random Forest classifier
emlearn_trees

Noise monitor

Microphone

Infinite Impulse Response filters
emlearn_iir

Image Classifier

Camera

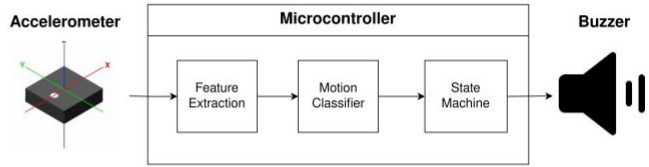
Examples for emlearn-micropython at
https://github.com/emlearn/emlearn-micropython#examples

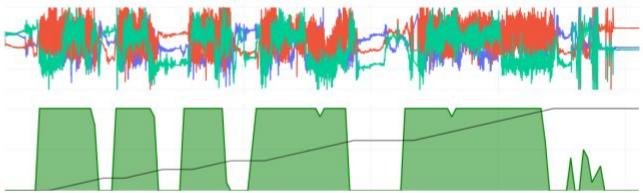
```
if is_MyCat(img):
    open_door()
```

Convolutional Neural Network
emlearn_cnn

Automatic toothbrush timer - what it does

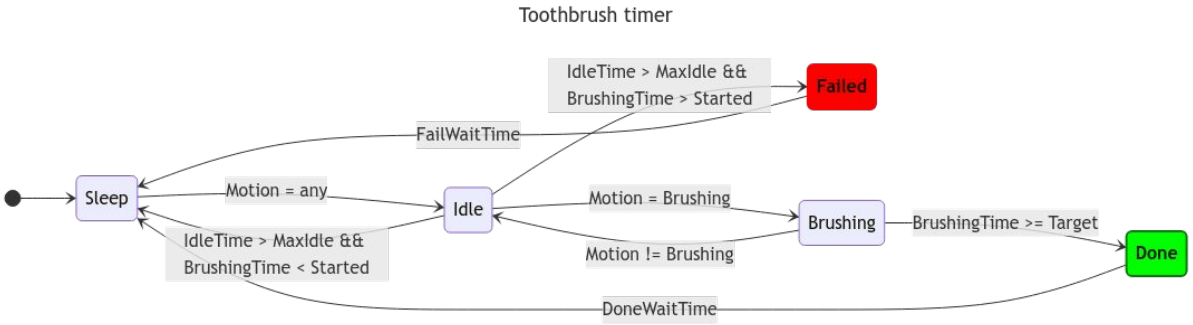
Detect when user is **actively brushing**
Sum up the **active time**
If hitting **2 minutes** - > Play **SUCCESS**
If **not completing** 2 minutes - > Play **FAIL**

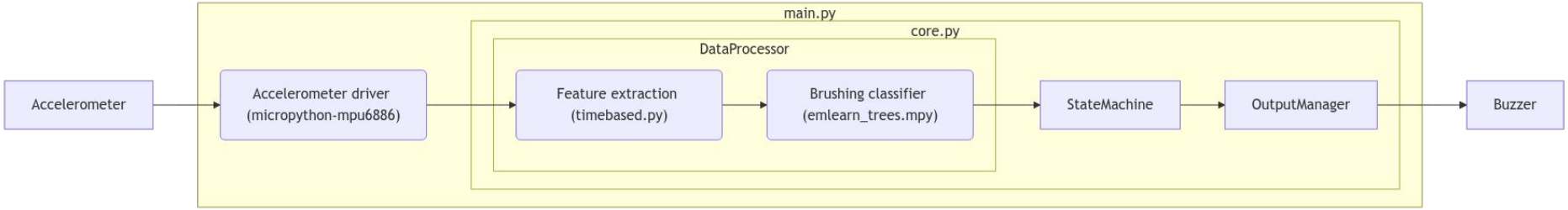




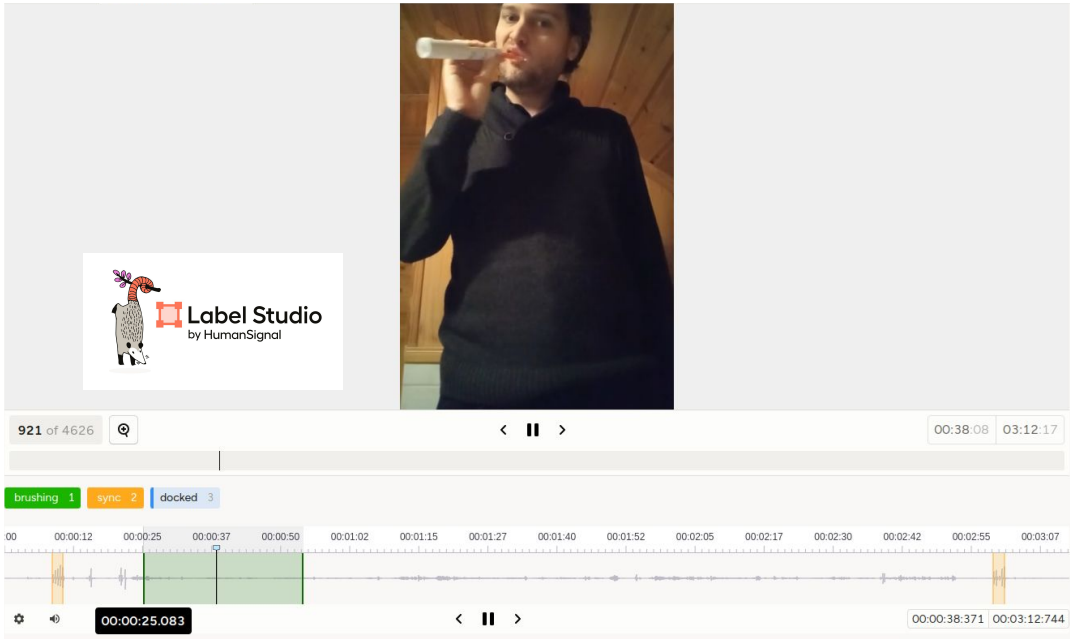
Hardware & firmware

M5StickC PLUS 2 from M5Stack
Around 500 lines of MicroPython





Data collection and labeling



Video recorded on phone
used as reference for labeling

start/end marked by tapping device
- used to sync video/labels from phone
with sensor data

Took around 1 hour

Using Label Studio for labeling

Data collection tools:
https://github.com/emlearn/emlearn-micropython/tree/master/examples/har_trees

Machine Learning pipeline

Using a scikit-learn based pipeline.

```
# Setup subject-based cross validation
splitter = GroupShuffleSplit(n_splits=n_splits, test_size=0.25,
    random_state=random_state)

# Random Forest classifier
clf = RandomForestClassifier(random_state = random_state,
    n_jobs=1, class_weight = "balanced")

# Hyper-parameter search
search = GridSearchCV(clf, param_grid=hyperparameters,
   scoring=metric, refit=metric, cv=splitter)
search.fit(X, Y, groups=groups)
```

```
toothbrush_jonnor': dict(
    groups=['session'],
    label_column = 'is_brushing',
    time_column = 'time',
    data_columns = ['x', 'y', 'z'],
    classes = [
        'True', 'False',
    ],
```

```
l = sorted(list(v))
l2 = [x*x for x in l]
sm = sum(l)
sqs = sum(l2)
avg = sum(l) / len(l)

median = l[MEDIAN]
q25 = l[Q1]
q75 = l[Q3]
iqr = (l[Q3] - l[Q1])

energy = ((sqs / len(l2)) ** 0.5)
std = ((sqs - avg * avg) ** 0.5)
```

```
(venv) [jon@jon-thinkpad har_trees]$ MIN_SAMPLES_LEAF=150,200,400 python har_train.py
-dataset har_exercise_1 --window-length 400 --window-hop 10
2024-12-04 12:54:52 [info     ] data-loaded                    dataset=har_exercise_1
uration=0.016095876693725586 samples=32000
2024-12-04 12:54:56 [info     ] feature-extraction-done        dataset=har_exercise_1
uration=4.534412145614624 labeled_instances=1952 total_instances=1952
```

```
Model written to ./har_exercise_1_trees.csv
Testdata written to ./har_exercise_1.testdata.npz
Results
    n_estimators  min_samples_leaf  mean_train_f1_micro  mean_test_f1_micro
0            10               150             0.996311            0.962705
1            10               200             0.995628            0.956557
2            10               400             0.986202            0.920902
```

```
import emlearn
converted = emlearn.convert(clf)
converted.save(name='gesture', format='csv', file='model.csv')
```

https://github.com/emlearn/emlearn-micropython/tree/master/examples/har_trees

Potential Improvements

- Easier install (no zipties)
- Reduce size
- Reduce costs

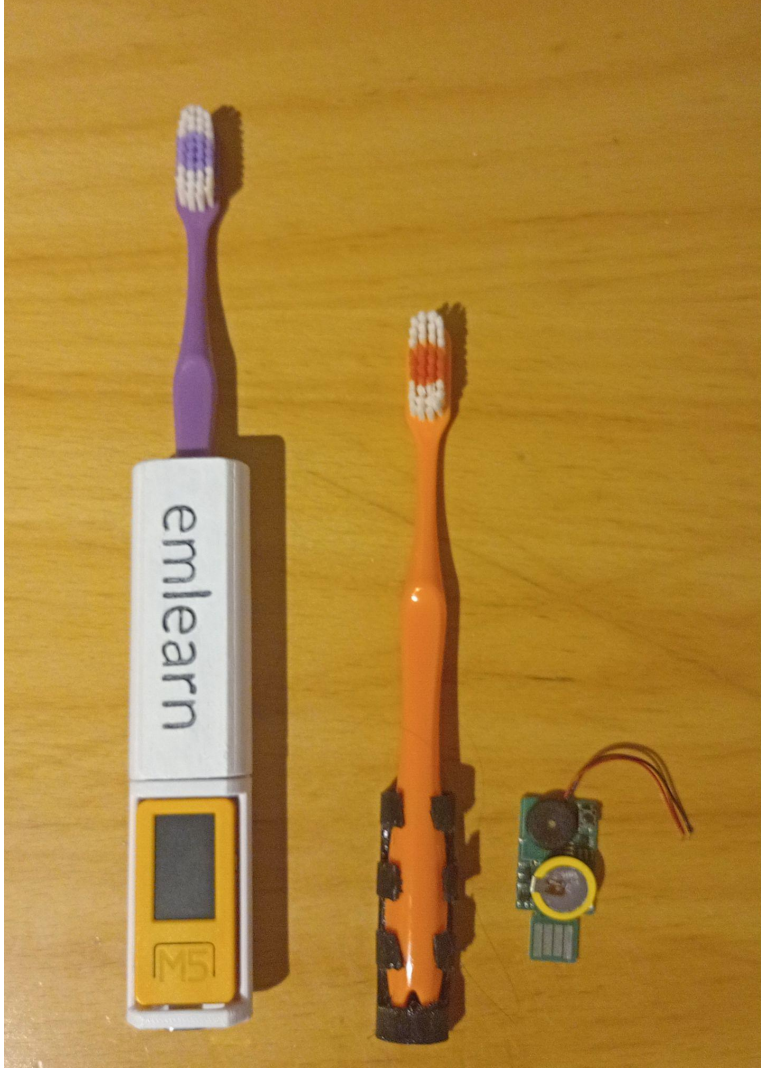
Flexible TPU casing

Custom PCB
Puya PY32 - 8 kB RAM / 64 kB FLASH

Using **emlearn C library**

https://hackaday.io/project/194511-1-dollar-tinyml

PCBWay



Conclusions

emlearn-micropython makes TinyML easy

- **MicroPython** means all application code can be in Python
- **Examples** give good starting points for practical use-cases
- **Custom models** can be possible with just a few hours of data collection

NB: also possible to use emlearn as a C library

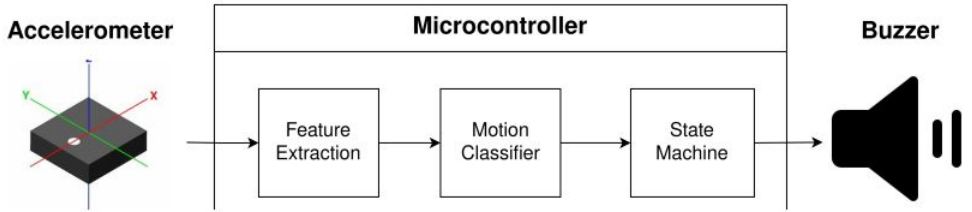
Get started

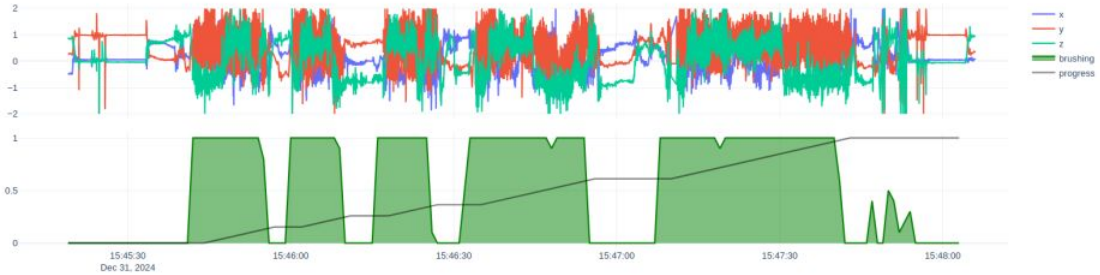
https://github.com/emlearn/emlearn-micropython

Automatic toothbrushing timer

Using accelerometer and machine learning

https://github.com/jonnor/toothbrush

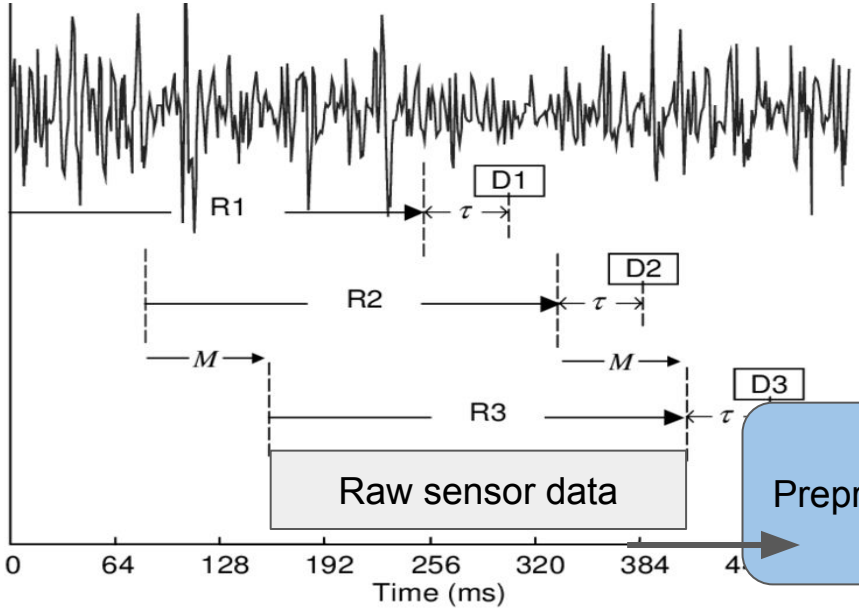




Powered by emlearn

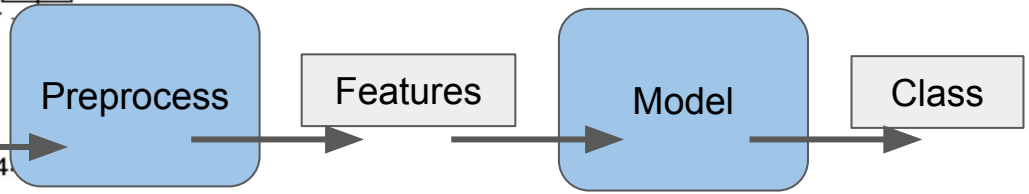
Bonus

ML on streams: Continuous classification



The sensor data stream
is sliced into overlapping windows.
Each window processed independently

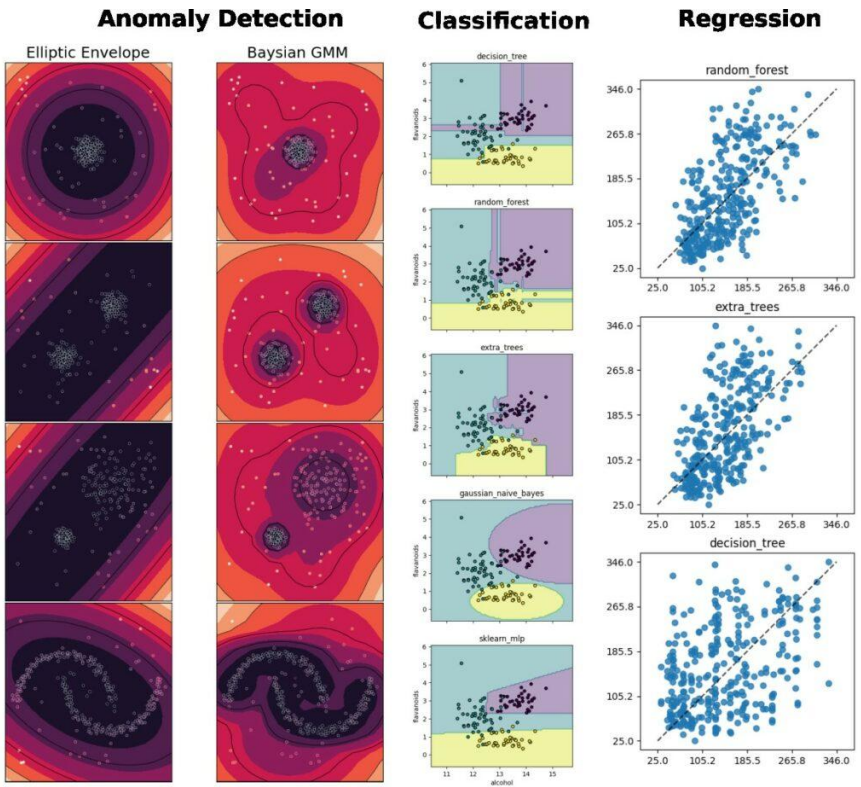
Example configuration
- 1 second window, every 1 second
- 50 Hz samplerate
- Processing time 30 ms



[42, 4002, … , 329]

“Brushing”

Supported tasks

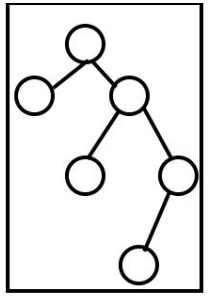


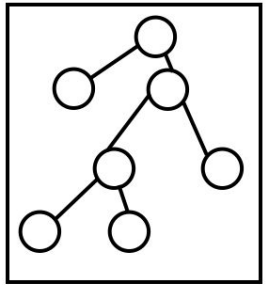
Supports the most common tasks for embedded
& sensor data use cases.

- Classification
- Regression
- Anomaly Detection

Supported models

Selection of simple & effective embedded-friendly models





emlearn

- Decision Trees (DT)
- Random Forest (RF)
- K Nearest Neighbors (KNN)
- Gaussian Mixture Models (GMM)
- Multi-Layer-Perceptron (MLP)
- Convolutional Neural Network (CNN)