MicroAI™ Atom Raspberry Pi Security EVK SDK 1.0

Contents

Disclaimer

Your use of the SDK is at your sole risk. You will be solely responsible for any damage to your computer system or loss of data that results from the download or use of the SDK. To the maximum extent permitted by applicable law, in no event shall ONE Tech or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including, without limitation damages for loss of business profits or revenue; business interruption or work stoppage; computer failure or malfunction; loss of business information, data or data use; loss of goodwill; or any other pecuniary loss) arising out of the use of or inability to use the SDK or the provision of or failure to provide support services, even if ONE Tech or its supplier has been advised of the possibility of such damages.

What is MicroAI™?

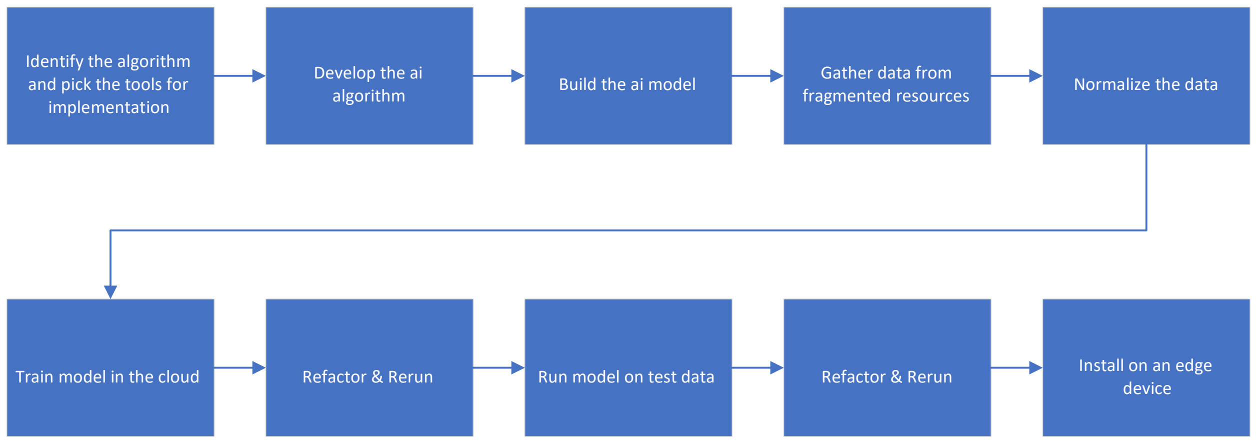
Simply put, MicroAI™ is an AI engine that can operate on low power edge devices. It can learn the pattern of any and all time series data and can be used to detect anomalies or abnormalities, make one step ahead predictions/forecasts.

Why is MicroAI™ good for your project

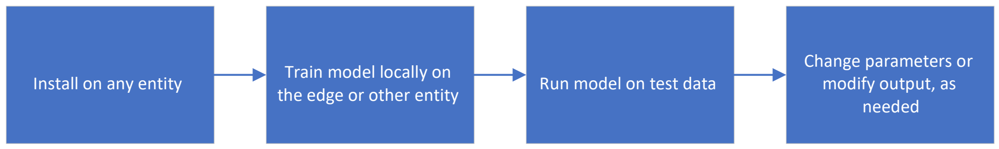
The MicroAI™ SDK enables your raspberry pi device or other entity with our proprietary AI (referred to in this document as 'MicroAI™') to help you build and gather feedback on your use cases, as well as the tutorials covered in this documentation. MicroAI™ can aid in the following tasks:

1. Data Collection
2. Data Organization
3. Model Building
4. Display, edit, and act upon the data gathered
5. Communicate with configured accessories and services to get them to perform actions, like turning on a light or receiving a notification on your phone.

Without MicroAI™ the process for building and integrating your own AI would work as follows:



Using MicroAI™ will save you development and cut down time to production significantly:


Install on any entity
Train model locally on the edge or other entity
Run model on test data
Change parameters or modify output, as needed

Understanding MicroAI™

The MicroAI™ SDK is made up of several key parts. The two most notable are the X-code and Y-code. However, all the parts can be broken down as follows:

Data Channels: Could be sensors, API's, Flat Files, or any avenue to receive time series data.

Data Acquisition(X-code): X-Code is better known as 'input'. One of the foundational pieces of MicroAI is that it is data source agnostic, meaning, that it can ingest data from a variety of sources. Using tools such as Redis and Numpy allows MicroAI to ingest sources such as, ambient temperature, gyroscope, and humidity data with a simple Python script.

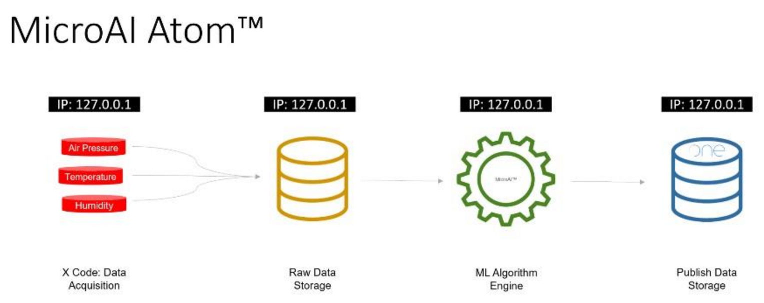
MicroAI: Is made up 3 steps: 1. Training the model 2. Engineering the features 3. Hosting the AI model. However, to define MicroAI simply, it provides insights into device behavior by comparing live data to comprehensive machine learning models.

Output Layer: Once the AI model has been trained and the inputs (X-code) has been configured for the necessary sensors, the MicroAI will begin sending data to the output layer. In this layer, MicroAI can be built up and/or customized to the user's needs.

Application Layer(Y-code): Y-Code is better known as 'output'. However, the application layer and the output layer are dependent on each other. The Application layer provides users flexibility of how they would like to deploy. Essentially, the application layer (Y-Code) can be created to run on top of the output layer for a customized experience. Examples include email alerts and local device behavior changes.

Visualization: This is exactly what it sounds like this can be in the form of dashboards and other outputs.

To visualize the relationship between these entities, please see the following diagram:



While the MicroAI™ libraries are proprietary (and encrypted), open source 'X-Code' (data acquisition) and 'Y-Code' (applications) can be written for multiple solutions.

This document covers MicroAI as implemented in the Security use case. Security is defined below:

Security: Simply put, security is the use of internal sensors within an entity or device to monitor activity and predict abnormalities in usage consistent with the likes of security breaches or vulnerabilities.

Requirements

Supported Devices

MicroAI™ can currently be supported on raspberry pi 3.

More devices will be added/recommended over time.

Tools & Versions

Once the MicroAI™ library is embedded in the device, you will need to make sure you have a few tools installed before we get started

- Python 3.7 and Sense Hat library (For APM only)
- Redis-server
- Make sure you have the latest version of Raspbian downloaded from the raspberry pi website (currently Raspbian buster lite)

Getting Started

Preparing your environment

Configuring your pi:

Regardless of if this is your first experience with using a raspberry pi or not, there are some things that need to be done to get you setup for ingesting the SDK.

After booting up your raspberry pi and attaching a monitor and keyboard, enter your username and password. After this, enter the following command:

`sudo raspi-config` then select ENTER

This should open the main menu.

Select number `7 Advanced Options` and select ENTER

Select A1 Expanding Space option and enable the expansion of space, after selecting ENTER you will need to reboot your raspberry pi for this change to take effect. Use the following command to reboot after exiting the main menu:

```
sudo reboot
```

Now you will need to enable Wi-Fi on your raspberry pi. To do this, inter the following command:

`sudo raspi-config` then select ENTER

Then in the main menu select `2 Network Options`

Then select `N2 Wi-Fi`

You will need to enter the wi-fi name and the password, then enable.

After enabling Wi-Fi, you will need to reboot your raspberry pi for this change to take effect. Use the following command to reboot after exiting the main menu:

```
sudo reboot
```

Once the raspberry pi has restarted, navigate back to the main menu using the same command as above to enable SSH. This will allow you to remotely access your raspberry pi via the command line or terminal. This will be incredibly helpful as we move into the next steps and you start trying the tutorials.

In the main menu, select number `5 Interfacing Options` and select ENTER

Then select `P2 SSH` and select ENTER. Select ENABLE. After enabling SSH you can exist the main menu and reboot if necessary.

Using SSH:

To after configuring SSH on your raspberry pi, simply open command line or terminal and type in the following command:

`ssh [raspberry pi username]@[IP address of your raspberry pi]` example:

```
ssh pi@123.456.7.890
```

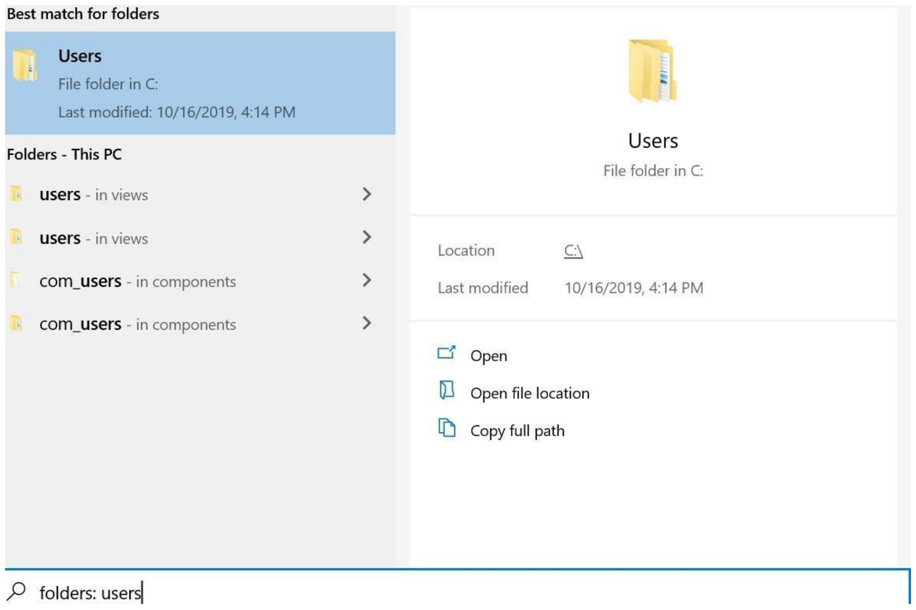
Then simply type in your password, select ENTER and you will have ssh remote access to your raspberry pi.

*To find the IP address of your raspberry pi simply type the following `ip -a` command: *

Possible Error Using SSH on a Windows PC:

If you are a user who has done this before and wanted to start again using a clean raspberry pi, your PC may remember the device (raspberry pi) and not let you access the raspberry pi (it should give an error and a warning of a possible ‘man in the middle’ attack).

To fix this, search for and open your Users folder.


Best match for folders

Users
File folder in C:
Last modified: 10/16/2019, 4:14 PM

Folders - This PC

users - in views

users - in views

com_users - in components

com_users - in components

Users
File folder in C:

Location C:\
Last modified 10/16/2019, 4:14 PM

Open
Open file location
Copy full path

folders: users

Then select your username > .ssh example: This PC >

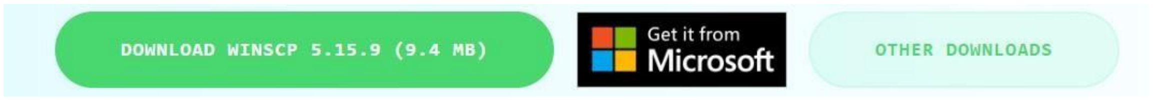
Windows (C:) > Users > john > .ssh

After navigating to your .ssh folder, delete the known_hosts file. This should remove your PC's memory of the device's IP address and you SSH into your raspberry pi again.

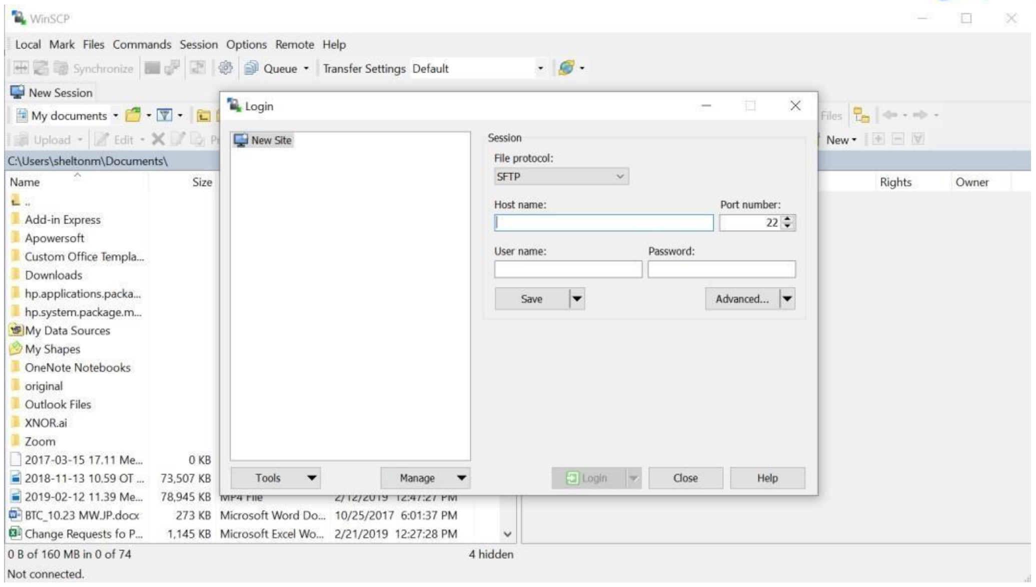
Getting the SDK:

Once your raspberry pi is up and running you will need to do a couple of things to make it a little easier to access and run your use cases. The process is as follows:

1. Go to the ONE Tech URL and download the SDK zip file
2. Extract this file
3. Download the WinSCP file manager here: https://winscp.net/eng/index.php this will help us move the necessary SDK files onto the raspberry pi



4. Once the download is completed open WinSCP and insert the raspberry pi IP address in the Host Name. In the Username and Password, simply type in the username and password for your raspberry pi.



File protocol: SFTP
Host name:
Port number: 22
User name:
Password:

```
C:\Users\sheltonm\Documents\
Name                         Size                                                      Rights   Owner
..
Add-in Express
Apowersoft
Custom Office Templa...
Downloads
hp.applications.packa...
hp.system.package.m...
My Data Sources
My Shapes
OneNote Notebooks
original
Outlook Files
XNOR.ai
Zoom
2017-03-15 17.11 Me...       0 KB
2018-11-13 10.59 OT ...  73,507 KB
2019-02-12 11.39 Me...   78,945 KB  MP4 File              2/12/2019  12:47:27 PM
BTC_10.23 MW.JP.docx        273 KB  Microsoft Word Do...  10/25/2017  6:01:37 PM
Change Requests fo P...   1,145 KB  Microsoft Excel Wo... 2/21/2019  12:27:28 PM
0 B of 160 MB in 0 of 74                                  4 hidden
Not connected.
```

5. Select Login. Once logged in, drag and newly downloaded and extracted SDK folder from your local computer to the raspberry pi (left to right). After you do this, you will be able to access the SDK files directly on your raspberry pi.

Raspberry Pi Prep:

Note: All the environment prep assets will need to be to be installed on the device (raspberry pi) directly. Any asset being installed on a device other than the raspberry pi (a laptop for instance) will be specifically identified.

The easiest way to get your hands on MicroAI™ and start building is prep your environment properly is by using pip. Be sure that the latest version of pip is installed (pip3), as you will be using this to install all other assets involved in the environment prep.

The first item is Python 3.7 (and above). After installation, we will need to install several python libraries. Now pop open terminal or command line, ssh into your raspberry pi and run the following:

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install python3-pip
4. sudo apt-get install sense-hat
5. sudo apt-get install redis-server
6. sudo reboot
7. pip3 install redis
8. pip3 install -Iv bottle==0.12.17
9. pip3 install -Iv tqdm==4.32.1

SDK Directory Structure

With the following SDK directory, you will find the following contents:

- **LICENSE_README.txt –** This will cover the legal license agreement you agree to by using MicroAI
- **MicroAI EVK SDK Version 1.1.pdf –** This is the manual you will use to guide you through the setup, installation, and tutorials of the SDK
- **SDK Docs –** explanations and and documentation for parts of the code.
- **Overview.txt –** explains at a high level what is in the SDK package

Within SDK Docs

- **Attack_Tool_README.txt –** will explain how to use the attack tool in IoT Security.
- **Attack_Tool_Requirements.txt –** will explain the versions of the 3rd party libraries referenced in the attack tool readme
- **microAI_Output_README.txt –** explains class for user created Y-Code

Projects > Security

- **IoT Attack Tool –** houses the command center, initiator, and the readme.md file ○ the command_center and initiator both house python scripts which are readable in command line or another IDE and the initiator houses the backdoor
- **Y –** houses the files required to view the output of MicroAI Network
- **X –** houses the files required to input data to MicroAI Network
- **AIengine –** houses the files required to train and execute MicroAI generated models

Projects > Security > X

- **signal_source -** begins the data ingestion for MicroAI Network (Security)
- **signalsource_router -** route table for the X-Code
- **rootcfg** – parameters that control details for data acquisition

Projects > Security > AI

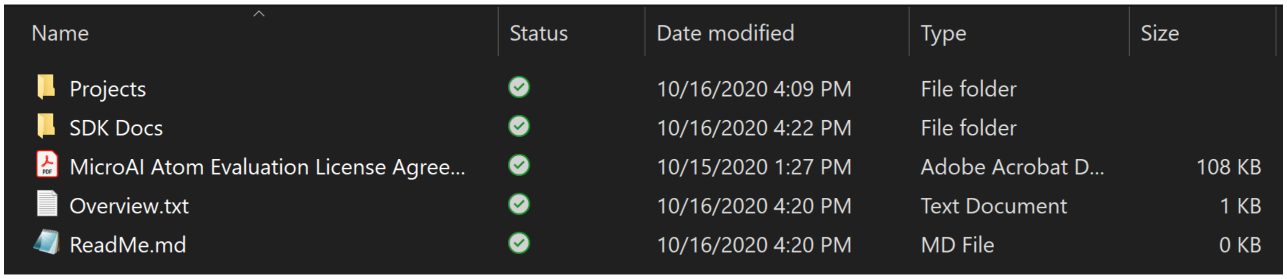
- **AIengine–** executable file that activates the training or execution of the AI generated model
- **AIengine_router–** route table for the AI engine
- **algPar–** Table of parameters that tune the execution of the model
- **rootcfg –** parameters that change the behavior of the training and execution processes
- **signalsource_router –** route table for the AI engine

Projects > Security > Y

- **AIengine_router–** route table for the Y code
- **basic_output.py–** monitors the output of MicroAI and prints if the groups are behaving normally or abnormally
- **MicroAI_Network_Output.cpython-37m-arm-linux-gnueabihf.so** - Encrypted class that allows access to methods for easy use of MicroAI Network's output

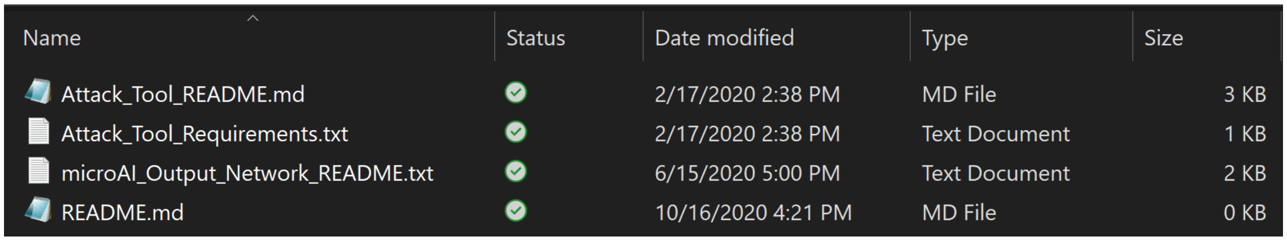
After downloading and extract the SDK zip file, the directory structure should look as follows:

Root:



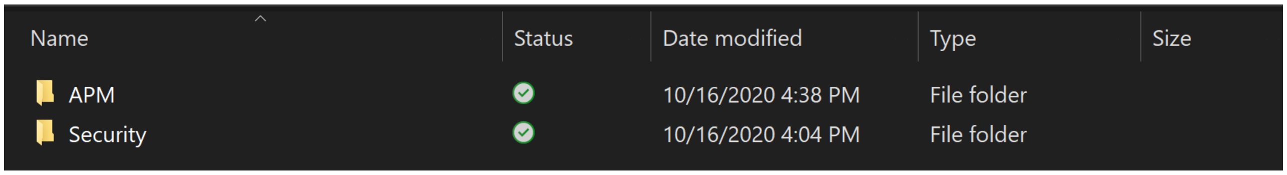
Name	Status	Date modified	Type	Size
Projects	✓	10/16/2020 4:09 PM	File folder	
SDK Docs	✓	10/16/2020 4:22 PM	File folder	
MicroAI Atom Evaluation License Agree...	✓	10/15/2020 1:27 PM	Adobe Acrobat D...	108 KB
Overview.txt	✓	10/16/2020 4:20 PM	Text Document	1 KB
ReadMe.md	✓	10/16/2020 4:20 PM	MD File	0 KB

Root > SDK_Docs:



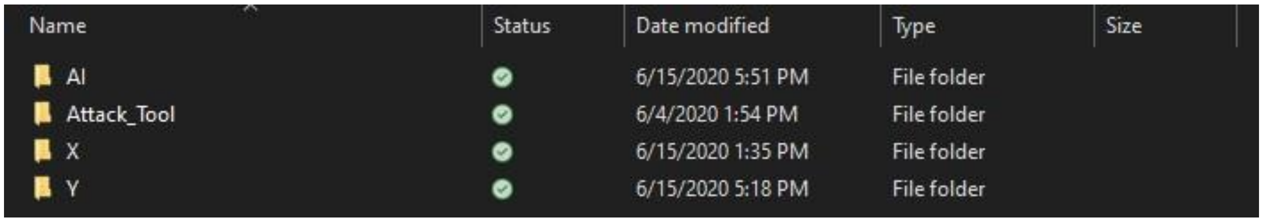
Name	Status	Date modified	Type	Size
Attack_Tool_README.md	✓	2/17/2020 2:38 PM	MD File	3 KB
Attack_Tool_Requirements.txt	✓	2/17/2020 2:38 PM	Text Document	1 KB
microAI_Output_Network_README.txt	✓	6/15/2020 5:00 PM	Text Document	2 KB
README.md	✓	10/16/2020 4:21 PM	MD File	0 KB

Root > Projects



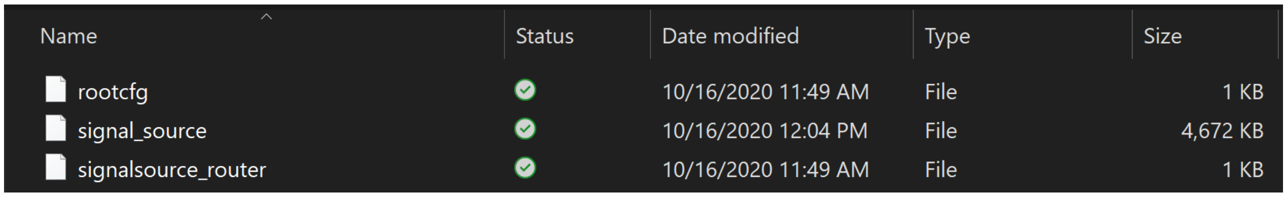
Name	Status	Date modified	Type	Size
APM	✓	10/16/2020 4:38 PM	File folder	
Security	✓	10/16/2020 4:04 PM	File folder	

Root > Projects > Security



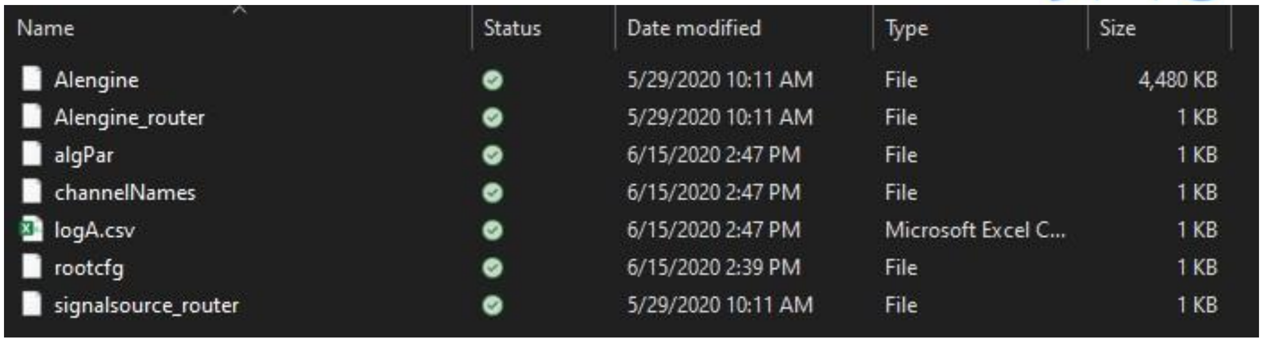
Name	Status	Date modified	Type	Size
AI	✓	6/15/2020 5:51 PM	File folder	
Attack_Tool	✓	6/4/2020 1:54 PM	File folder	
X	✓	6/15/2020 1:35 PM	File folder	
Y	✓	6/15/2020 5:18 PM	File folder	

Root > Projects > Security > X



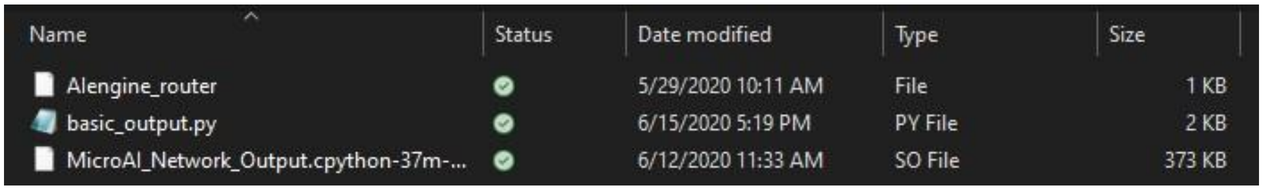
Name	Status	Date modified	Type	Size
rootcfg	✓	10/16/2020 11:49 AM	File	1 KB
signal_source	✓	10/16/2020 12:04 PM	File	4,672 KB
signalsource_router	✓	10/16/2020 11:49 AM	File	1 KB

Root > Projects > Security > AI



Name	Status	Date modified	Type	Size
AIengine	✓	5/29/2020 10:11 AM	File	4,480 KB
AIengine_router	✓	5/29/2020 10:11 AM	File	1 KB
algPar	✓	6/15/2020 2:47 PM	File	1 KB
channelNames	✓	6/15/2020 2:47 PM	File	1 KB
logA.csv	✓	6/15/2020 2:47 PM	Microsoft Excel C...	1 KB
rootcfg	✓	6/15/2020 2:39 PM	File	1 KB
signalsource_router	✓	5/29/2020 10:11 AM	File	1 KB

Root > MicroAI_SDK1.1 > Security > Y



Name	Status	Date modified	Type	Size
AIengine_router	✓	5/29/2020 10:11 AM	File	1 KB
basic_output.py	✓	6/15/2020 5:19 PM	PY File	2 KB
MicroAI_Network_Output.cpython-37m-...	✓	6/12/2020 11:33 AM	SO File	373 KB

IoT Security

Configuring your X-Code – Security

Because the specific input channels used in the IoT Security use case are kept hidden. An executable has been provided that is already preconfigured for this use case. The only thing the user must do is check the router tables in the X, AI, and Y directories to make sure they all match. A sample of the router tables can be seen below.

```
channelID,IP0,IP1,IP2,IP3,DB1,DB3,groupID
c0,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
c1,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
c2,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
c3,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
c4,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
c5,127.0.0.1,127.0.0.1:6379,127.0.0.1,127.0.0.1:6379,0,0,1
```

Column 1 is the name of the channel name. This is the name the AI engine will use to identify each channel

Column2 is the IP address of the input signal. The data ingestion code should always check this value to ensure that is either "localhost", "127.0.0.1" or the local IP of the device that is running the data ingestion or X code. If it is not either of those three values, the X-Code should skip this channel

Colmun3 is the IP address and port number of the redis server where that channel's input code will be stored

Column4 is the IP address of the MicroAI engine that will be processing the data

Column5 is the IP address and port number of the redis server where the output of the AI engine will be stored

Column6 and 7 are currently unused

Column8 is the group number for the data. This allows the processing of the data to be managed. The output of each channel is partially dependent on all other channels within the same group.

For this example, all channels will be given local addresses as everything will be run and stored on this single device.

How to use it

Training & Running the Model

Before the MicroAI™ engine can be run, it must first construct a training model. The more data points the training model has, the more accurate it should be. It should be noted that having more data points does not increase the size of the model but does increase the accuracy. First let us leave the X-Code directory and enter the AIengine directory

```
cd..
```

```
cd AI/
```

To begin training the model, open rootcfg.

```
nano rootcfg
```

First ensure that 5ifBuildModel is 1 and 6ifExecuteModel is 0. Next we will edit 3hDim and 4tDim. 4tDim is the number of data points that will be used to train the model. By default, it should be 200. 3hDim should be set to be 1/10th the size of 4tDim. So, for our case, we shall set it to 20. Save the file and exit back to the terminal.

Next we will train the model by running

```
./AIengine
```

We will now wait for the AI to finish training. This should take approximately 4tDim * 16samplingTime milliseconds. So, in our case, it should take around 400 seconds or roughly five and a half minutes. Once this process is complete, we need to edit rootcfg again

```
nano rootcfg
```

This time we will set 5ifBuildModel to 0 and 6ifExecuteModel to 1. The next time we activate the AIengine executable, it will not train the model, but instead activate the AI.

```
./AIengine
```

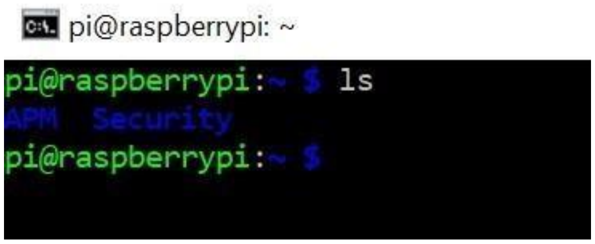
IoT Security Tutorial – Step 1

(see pg. 4 for definition of Security)

By default, these tutorials will be preloaded in the SDK. To begin, open command prompt, ssh into your raspberry pi, and type the following command:

```
ls
```

You should see two directories: APM and Security



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ ls
APM  Security
pi@raspberrypi:~ $
```

You should then cd into the directory/tutorial you want to run through by typing the following command:

```
cd Security
```

After 'cd' type the name of the directory (APM or Security)

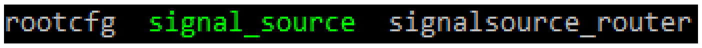
Step 2

To run MicroAI IoT Security Network. First go to MicroAI/Security/X with cd command.

After switching into the Security directory type the following command:

```
ls
```

You should see the following file structure:



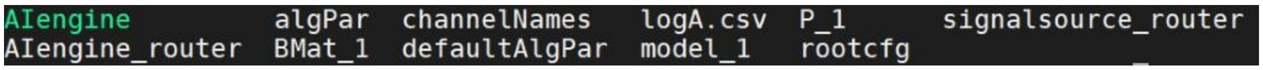
```
rootcfg  signal_source  signalsource_router
```

Use the following command to run X code.

```
./signal_source
```

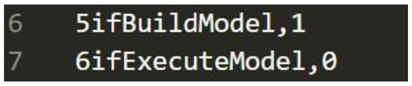
Next, open another terminal, and navigate to MicroAI/Security/AI.

Using 'ls' command, you will see the following files.



```
AIengine         algPar  channelNames   logA.csv  P_1      signalsource_router
AIengine_router  BMat_1  defaultAlgPar  model_1   rootcfg
```

Open ‘rootcfg’ with any text editor you like. Change the value of ‘5ifBuildModel’ to 1, and the value of ‘6ifExecutaModel’ to 0. This will set MicroAI to training mode.



```
5ifBuildModel,1
6ifExecuteModel,0
```

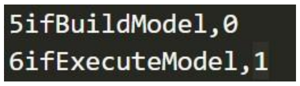
Then, execute MicroAI using the following command:

```
./AIengine
```

If you run into permission error, please change the permission of this file to allow execution.

```
sudo chmod 775 AIengine
```

Wait for MicroAI to finish training. When it’s done, Open ‘roofcfg’ again, and change the value of ‘5ifBuildModel’ to 0, and the value of ‘6ifExecutaModel’ to 1. This will set MicroAI to execution mode.

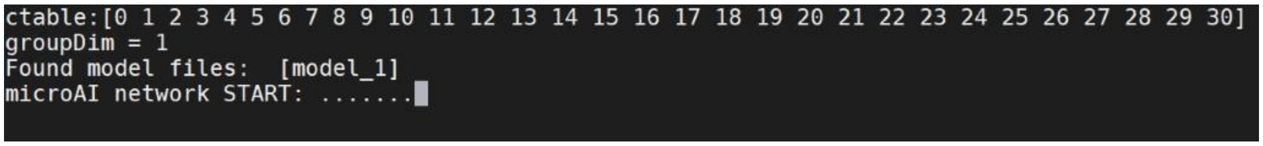


```
5ifBuildModel,0
6ifExecuteModel,1
```

Now you can execute MicroAI.

```
./AIengine
```

You’ll see some output look like this.



```
ctable:[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30]
groupDim = 1
Found model files:  [model_1]
microAI network START: .......
```

The engine is now up and running. Open another terminal. Let’s run a simple application script or YCode to see its output. First Navigate to the Y-Code directory.

From here you must ensure that the route table in this directory is identical to the ones in the previously discussed directories.

Activate the Y-Code.

```
python3 basic_output.py
```

This script will loop forever as it prints the behavior of each group within the MicroAI every 2 seconds.

Limitations and Dependencies

- Currently, the only accepted device for running MicroAI in the raspberry pi 3. However, you can also run MicroAI on a virtual machine(VM) in some use cases. Before running MicroAI on a VM, the user should note that fairly this is use case is for advanced users experienced in trouble shoot both database and other programming related issues.
- Data will not be directly stored.
- Future versions of packages and devices may not be supported. Additional documentation and doc updates will be given to accommodate updates.
- This is an evaluation version only and should not be expected to support a full production environment.

- **Max channel numbers = 6**
- **Max training entries = 3600**
- **Max execution entries = 129600 (roughly fifteen days of continuous execution)**

Reference Materials

- For questions or to get help please post your questions here: http://developers.ONE Tech.ai/community/

Can I use other sensors on my raspberry pi 3?
Yes, other sensors are accepted, however, the one documented here is a sense HAT.

Can I modify MicroAI™?
No. You may only modify X-code and Y-code. Which is essentially the input and output.

What are the requirements for running MicroAI™ on Windows?
MicroAI™ uses Python version 3.7.3 with these dependencies: NumPy, Psutil, waitress, dash, bottle, tablib, and paho-mqtt. For Redis, version 2.4 and above is sufficient.

Is there going to be a GUI for the MicroAI™ configuration, or will it always be command line/terminal based?
MicroAI™ is currently for use by developers in command line/terminal. No GUI is on the roadmap currently.

Is MicroAI™ supervised or unsupervised learning?
MicroAI™ uses semi-supervised learning. Model training does occur; however, it also learns on its own, in real time.

After you download the MicroAI™ SDK on your computer, do you need to run any installation?
No, once you have the file unzipped and your python dependencies installed, you can get started with MicroAI™.

If connecting to the device using SSH, do users need a VPN into a certain network for MicroAI™ to work?
No, the machine they are SSHing from just needs to be on the same network as the device they are running MicroAI™ on.

If I launch an attack simulation right after an attack has just completed, will the AI detect this?
It depends. However, it is probably best to wait 20 minutes before running another attack script so the device behavior can settle.

How far ahead of time can MicroAI™ forecast?
MicroAI™ predicts values one step ahead of the current value. This could be one second ahead or hours ahead based on the output frequency.

Will MicroAI™ be able to detect anomalies that happen over long periods of time?
This will depend on how long the AI model is trained, as well as, the frequency of change.

What happens if the IoT Security AI model is trained while an attack is occurring or while a backdoor has already been exploited?
It is pertinent that when the MicroAI™ model is being trained, there are no previous or currently exploited vulnerabilities. These will make the predictive analysis inaccurate.

For training, can MicroAI™ intake data that is not part of real time asset data? Such as industry standard levels over time or lifetime data of a similar asset?
Yes, MicroAI™ can intake historical data as part of the training dataset.

Can different parameters fed into MicroAI™ be given different weights?

Yes, using feature engineering capabilities, MicroAI™ can incorporate different weights of channel values for optimal predictive analysis.

Can IoT security detect any vulnerabilities that are being exploited outside of the device?

When running on an edge device, MicroAI™ can only detect vulnerabilities on the device level. Not the network and application levels.

What is the limit of devices or channels that can be on a single MicroAI™ deployment?

Currently, MicroAI™ supports 100’s of channels being ingested from a single edge device. However, the EVK limits users to 6 channels.

Permission Denied. How do I get execution rights to an executable file?

Some users will experience a permission denied error when attempting to run the executable files in the demos. To remedy this, use the command `chmod +x executable_name_here`.