A picture containing text, electronics

Description automatically generated

[OpenMV Bluetooth TripWire (EN)](https://www.youtube.com/watch?v=gCpC2uOva90)

Hi if you haven’t watched the demo video, feel free the play this  short clip to see what it’s capable of.

**Introduciton**

This project took the inspiration from the another open-source project **daytripper** ([link 1](https://github.com/dekuNukem/daytripper)) which uses 2 seperate devices to detect movement and control your PC to switch Apps. However, I think if we go with the Computer Vision solution instead, we might reduce the number of hardware to just 1, and we can even push it a little further by adding some more cool features like face recognition, speed detection and even more.

That’s how I came up with this idea – using **OpenMV**, which is littler and easier to deploy, and a **IoT Microcontroller**, in this case Ameba RTL8722DM\_MINI, together we can achieve the same function as **daytripper** and more.

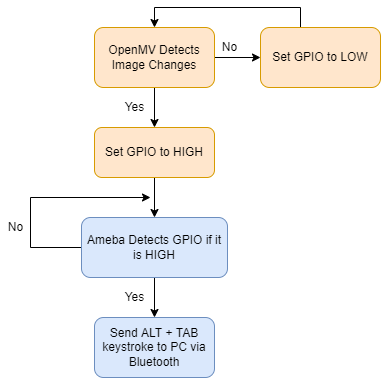
**Components**

1. Ameba RTL8722DM\_MINI dev board x1
2. OpenMV(any model) dev board x1

**Connection** is simple, just connect P0 pin on OpenMV to pin 9 on Ameba

**Function Flow**

This is how it works, 



**Code**

**OpenMV**

*# Advanced Frame Differencing Example*

*#*

*# This example demonstrates using frame differencing with your OpenMV Cam. This*

*# example is advanced because it preforms a background update to deal with the*

*# backgound image changing overtime.*

import sensor, image, pyb, os, time

**from** pyb import Pin

p\_out = Pin('P0', Pin.OUT\_PP)

p\_out.low()

TRIGGER\_THRESHOLD = 5

BG\_UPDATE\_FRAMES = 50 *# How many frames before blending.*

BG\_UPDATE\_BLEND = 128 *# How much to blend by... ([0-256]==[0.0-1.0]).*

sensor.reset() *# Initialize the camera sensor.*

sensor.set\_pixformat(sensor.RGB565) *# or sensor.RGB565*

sensor.set\_framesize(sensor.QVGA) *# or sensor.QQVGA (or others)*

sensor.skip\_frames(time = 2000) *# Let new settings take affect.*

sensor.set\_auto\_whitebal(False) *# Turn off white balance.*

clock = time.clock() *# Tracks FPS.*

*# Take from the main frame buffer's RAM to allocate a second frame buffer.*

*# There's a lot more RAM in the frame buffer than in the MicroPython heap.*

*# However, after doing this you have a lot less RAM for some algorithms...*

*# So, be aware that it's a lot easier to get out of RAM issues now. However,*

*# frame differencing doesn't use a lot of the extra space in the frame buffer.*

*# But, things like AprilTags do and won't work if you do this...*

extra\_fb = sensor.alloc\_extra\_fb(sensor.width(), sensor.height(), sensor.RGB565)

**print**("About to save background image...")

sensor.skip\_frames(time = 2000) *# Give the user time to get ready.*

extra\_fb.replace(sensor.snapshot())

**print**("Saved background image - Now frame differencing!")

triggered = False

frame\_count = 0

**while**(True):

clock.tick() *# Track elapsed milliseconds between snapshots().*

img = sensor.snapshot() *# Take a picture and return the image.*

frame\_count += 1

**if** (frame\_count > BG\_UPDATE\_FRAMES):

frame\_count = 0

*# Blend in new frame. We're doing 256-alpha here because we want to*

*# blend the new frame into the backgound. Not the background into the*

*# new frame which would be just alpha. Blend replaces each pixel by*

*# ((NEW\*(alpha))+(OLD\*(256-alpha)))/256. So, a low alpha results in*

*# low blending of the new image while a high alpha results in high*

*# blending of the new image. We need to reverse that for this update.*

img.blend(extra\_fb, alpha=(256-BG\_UPDATE\_BLEND))

extra\_fb.replace(img)

*# Replace the image with the "abs(NEW-OLD)" frame difference.*

img.difference(extra\_fb)

hist = img.get\_histogram()

*# This code below works by comparing the 99th percentile value (e.g. the*

*# non-outlier max value against the 90th percentile value (e.g. a non-max*

*# value. The difference between the two values will grow as the difference*

*# image seems more pixels change.*

diff = hist.get\_percentile(0.99).l\_value() - hist.get\_percentile(0.98).l\_value()

triggered = diff > TRIGGER\_THRESHOLD

**if** triggered == True:

p\_out.high()

**else**:

p\_out.low()

**print**(clock.fps(), triggered) *# Note: Your OpenMV Cam runs about half as fast while*

*# connected to your computer. The FPS should increase once disconnected.*

**Ameba**

#include "BLEHIDDevice.h"

#include "BLEHIDKeyboard.h"

#include "BLEDevice.h"

BLEHIDKeyboard keyboardDev;

BLEAdvertData advdata;

#define ENABLE\_PIN 9

**void** **setup**() {

Serial.begin(115200);

advdata.addFlags();

advdata.addCompleteName("AMEBA\_BLE\_HID");

advdata.addAppearance(GAP\_GATT\_APPEARANCE\_HUMAN\_INTERFACE\_DEVICE);

advdata.addCompleteServices(BLEUUID(HID\_SERVICE\_UUID));

BLEHIDDev.init();

BLE.init();

BLE.configAdvert()->setAdvData(advdata);

BLE.setDeviceName("AMEBA\_BLE\_HID");

BLE.setDeviceAppearance(GAP\_GATT\_APPEARANCE\_HUMAN\_INTERFACE\_DEVICE);

BLE.configSecurity()->setPairable(true);

BLE.configSecurity()->setAuthFlags(GAP\_AUTHEN\_BIT\_BONDING\_FLAG);

BLE.configServer(3);

BLE.addService(BLEHIDDev.hidService());

BLE.addService(BLEHIDDev.battService());

BLE.addService(BLEHIDDev.devInfoService());

pinMode(ENABLE\_PIN, INPUT);

BLE.beginPeripheral();

}

**int** flag = 0;

**void** **loop**() {

**if** (BLE.connected() && digitalRead(ENABLE\_PIN) && flag == 0) {

Serial.println("Sending keystrokes");

keyboardDev.keyReleaseAll();

delay(100);

keyboardDev.keyPress(HID\_KEY\_ALT\_LEFT);

delay(100);

keyboardDev.keyPress(HID\_KEY\_TAB);

keyboardDev.keyReleaseAll();

delay(100);

flag = 1;

} **else** {

flag = 0;

delay(100);

}

}

**Conclusion**

This project is not perfect as it’s done in a rush, so if anyone wants to perfect it you may go ahead and change my code however you like, or leave a comment below if you have a question or want to discuss something with me~

Until next time, happy coding 