/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Built for use with the www.rallyblitz.com apps. Built with the help of

 Adafruit code samples and other codes samples.

 Known to work with Bluefruit Feather 32u4 but should work with the

 feather M0

 Support Adafruit and the Arduino communities. Their products and code

 samples helped make this work.

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

 Functions to be added

 - Timeout function

 - LED controls

 - More complete power management for battery consumption

 - Add more pinout and adjust cycle times

\*/

#include <Arduino.h>

#include <SPI.h>

#if not defined (\_VARIANT\_ARDUINO\_DUE\_X\_) && not defined(ARDUINO\_ARCH\_SAMD)

 #include <SoftwareSerial.h>

#endif

#include "Adafruit\_BLE.h"

#include "Adafruit\_BluefruitLE\_SPI.h"

#include "Adafruit\_BluefruitLE\_UART.h"

#include "BluefruitConfig.h"

/\*=========================================================================

 APPLICATION SETTINGS

    FACTORYRESET\_ENABLE    Perform a factory reset when running this sketch

    Enabling this will put your Bluefruit LE module

 in a 'known good' state and clear any config

 data set in previous sketches or projects, so

    running this at least once is a good idea.

    When deploying your project, however, you will

 want to disable factory reset by setting this

 value to 0.  If you are making changes to your

    Bluefruit LE device via AT commands, and those

 changes aren't persisting across resets, this

 is the reason why.  Factory reset will erase

 the non-volatile memory where config data is

 stored, setting it back to factory default

 values.

    Some sketches that require you to bond to a

 central device (HID mouse, keyboard, etc.)

 won't work at all with this feature enabled

 since the factory reset will clear all of the

 bonding data stored on the chip, meaning the

 central device won't be able to reconnect.

 MINIMUM\_FIRMWARE\_VERSION Minimum firmware version to have some new features

 -----------------------------------------------------------------------\*/

 #define FACTORYRESET\_ENABLE 1

/\*=========================================================================\*/

/\* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST \*/

Adafruit\_BluefruitLE\_SPI ble(BLUEFRUIT\_SPI\_CS, BLUEFRUIT\_SPI\_IRQ, BLUEFRUIT\_SPI\_RST);

// A small helper

void error(const \_\_FlashStringHelper\*err) {

 Serial.println(err);

 while (1);

}

typedef struct

{

 uint8\_t modifier; /\*\*< Keyboard modifier keys \*/

 uint8\_t reserved; /\*\*< Reserved for OEM use, always set to 0. \*/

 uint8\_t keycode[6]; /\*\*< Key codes of the currently pressed keys. \*/

} hid\_keyboard\_report\_t;

// Report that send to Central every scanning period

hid\_keyboard\_report\_t keyReport = { 0, 0, { 0 } };

// Report sent previously. This is used to prevent sending the same report over time.

// Notes: HID Central intepretes no new report as no changes, which is the same as

// sending very same report multiple times. This will help to reduce traffic especially

// when most of the time there is no keys pressed.

// - Init to different with keyReport

hid\_keyboard\_report\_t previousReport = { 0, 0, { 1 } };

// GPIO corresponding to HID keycode

int inputPins[6] = { 5 , 6 , 9 , 10 , 11 , 12 };

char const \*commands[6] = { "VOLUME+", "VOLUME-", "MEDIANEXT", "MEDIAPREVIOUS", "VOLUME+", "VOLUME-" };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

 @brief Sets up the HW an the BLE module (this function is called

 automatically on startup)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup(void)

{

 //while (!Serial);

 delay(500);

 Serial.begin(115200);

 Serial.println(F("RallyBlitz Button Control"));

 Serial.println(F("---------------------------------------"));

 /\* Initialise the module \*/

 Serial.print(F("Initialising the Bluefruit LE module: "));

 if ( !ble.begin(VERBOSE\_MODE) )

 {

 error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));

 }

 Serial.println( F("OK!") );

 if ( FACTORYRESET\_ENABLE )

 {

 /\* Perform a factory reset to make sure everything is in a known state \*/

 Serial.println(F("Performing a factory reset: "));

 ble.factoryReset();

 }

 /\* Disable command echo from Bluefruit \*/

 ble.echo(false);

 Serial.println("Requesting Bluefruit info:");

 /\* Print Bluefruit information \*/

 ble.info();

 /\* Enable HID Service if not enabled \*/

 int32\_t hid\_en = 0;

 ble.sendCommandWithIntReply( F("AT+BleHIDEn"), &hid\_en);

 if ( !hid\_en )

 {

 Serial.println(F("Enable HID Service (including Keyboard): "));

 ble.sendCommandCheckOK(F( "AT+BleHIDEn=On" ));

 /\* Add or remove service requires a reset \*/

 Serial.println(F("Performing a SW reset (service changes require a reset): "));

 !ble.reset();

 }

 Serial.println();

 Serial.println(F("Go to your phone's Bluetooth settings to pair your device"));

 Serial.println(F("then open an application that accepts keyboard input"));

 Serial.println();

 // Set device name

ble.println("AT+GAPDEVNAME=RallyBlitz Control");

ble.println("ATZ");

 // Set up input Pins

 for(int i=0; i< 6; i++)

 {

 pinMode(inputPins[i], INPUT\_PULLUP);

 }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*!

 Constant poll for new button press

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop(void)

{

 /\* scan all GPIO listed in definition

 \*/

 if ( ble.isConnected() )

 {

 int activePinIndex = 0;

 for(int i=0; i<6; i++)

 {

 // GPIO / Button is a press on ground signal

 if ( digitalRead(inputPins[i]) == LOW )

 // {

 // keyReport.keycode[i] = inputKeycodes[i];

 // activePinIndex = i;

 // }else

 // {

 // keyReport.keycode[i] = 0;

 // }

 {

 // Send keyboard report through BLE

 ble.print("AT+BleHidControlKey=");

 ble.println(commands[i]);

 }

 }

 }

 // scaning period is 175 ms

 delay(175);

}