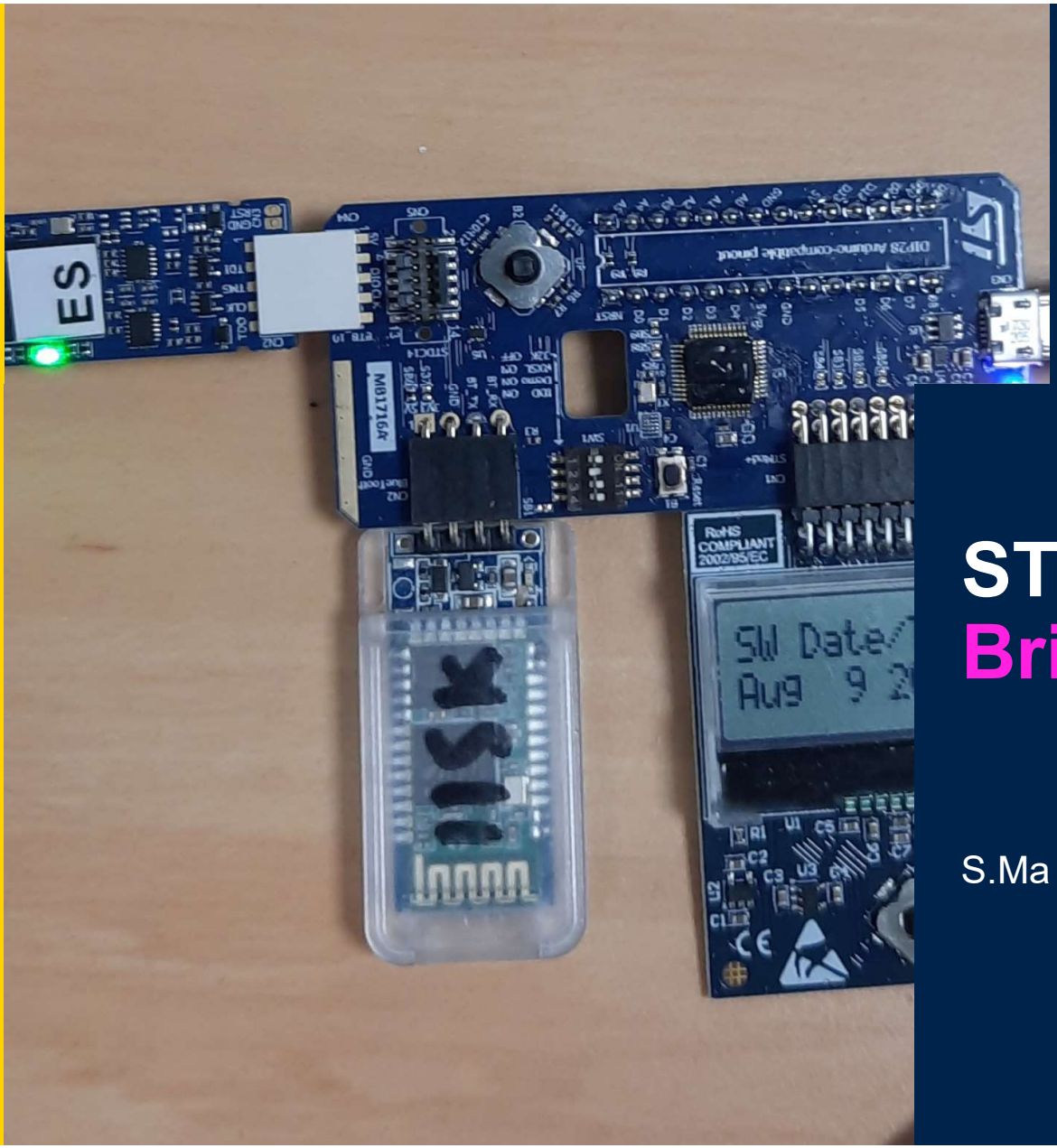


life.augmented

STM32C0316-DK Demo
Brisk application subset

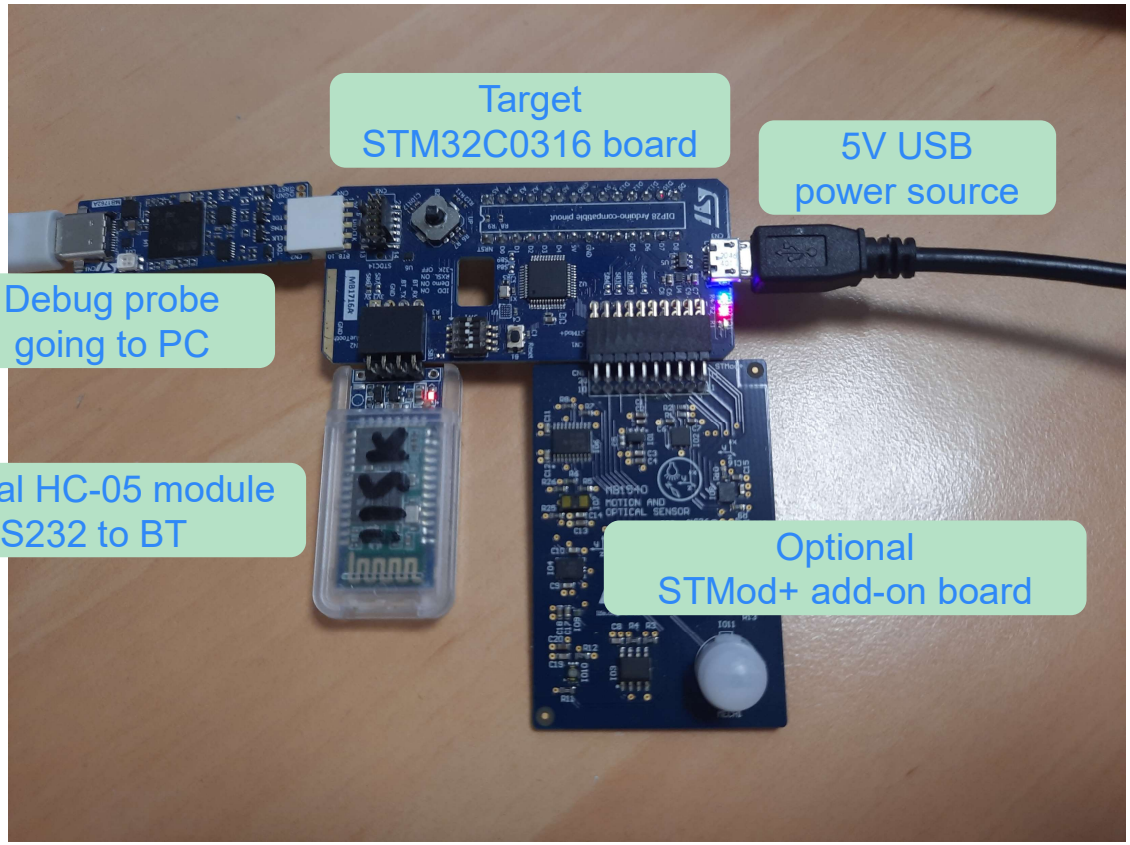
S.Ma

Overview

- Brisk was an internal demoset originally built on STM32L496/L4R
  - It is built from an assembly of various application blocks
  - This demo was reused and stripped down to fit on STM32C0316-DK, first using IAR, and CubeIDE 1.10.0 in 2022
  - Source code was developed aiming to be intuitive/reusable/resource-light/handy for board bring-up
  - It could further be polished up and use some calibration tricks to be core frequency and compile option independent
  - It uses physical units (the human world's) mV, msec, uS, degC_x10
  - It tries to choose the right balance between code efficiency, ease of taking ownership, and modular

- One goal is to be able to go back to the code a year later, and quickly be able to get back to coding

How to play with Brisk demo ?



- Install CubeIDE for STM32C0
- Install and build the Brisk Demo
- Provide +5V power by microUSB
- Plug debug probe and connect to PC
- No need to plug an STMod+ add-on

- Optional: Plug HC-05 dongle
  - Dongle setting: 115200bps,8,n,1
  - Its red LED should blink
  - Set the slide switch as in the picture
- Go to Debug mode and run the Brisk app
  - Green LED should pulse beat slowly

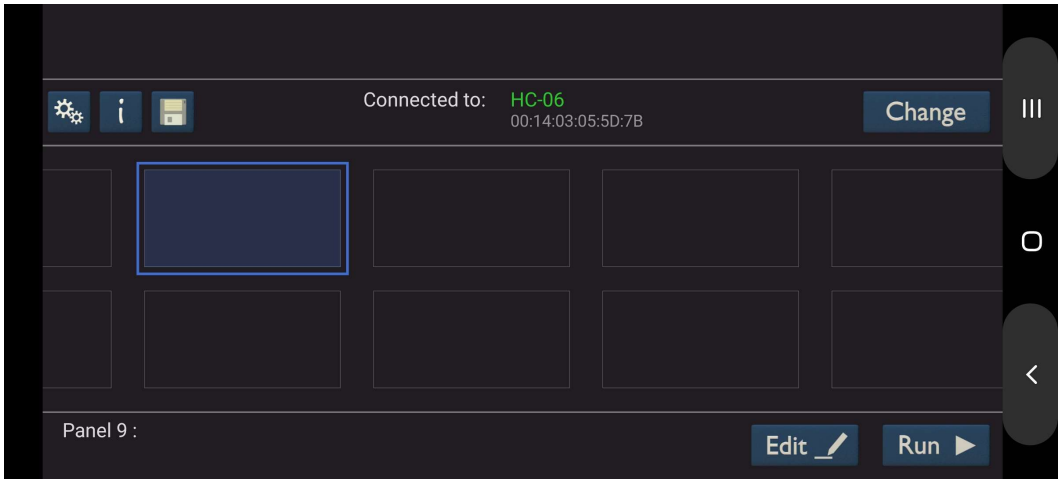
life.augmented

How to play with Brisk demo ?

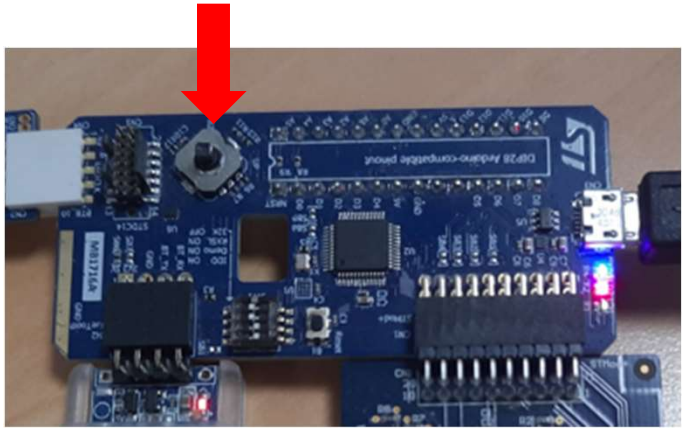
- Run the Android BTEL app
- Select “Connect” while HC-05 is on
- Discover the dongle
  - Its address maybe 00:14:03:05:xx:xx
  - Its name may vary
  - Select and connect
  - HC-05 Red led stops blinking and stays ON
- Click Done

life.augmented

How to play with Brisk demo ?



- Select one empty dashboard
- Click “Run”
- Keep pushing down the key (5~6 sec)
- A live dashboard will pop up



life.augmented

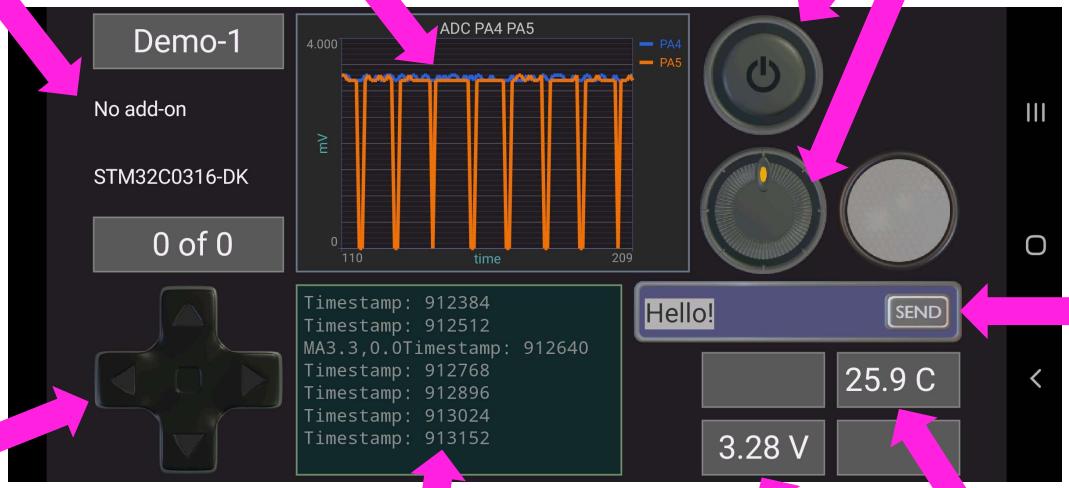
How to play with Brisk demo ?

2/2

Oscilloscope view of PA4 and PA5 ADC channels

Show if any add-on board is detected

Take control of the green LED dimming
Adjust the dimming level from the phone



Send Text to STM32 other UART

When an STMod+ add-on board is present,
enables vertical dashboard navigation
(the right displayed part changes)

STM32 “printf” to show timestamp
And show demo is alive

STM32 Vdd ADC measured voltage

STM32 ADC measured Temperature
(put your finger on chip, use hair dryer….)

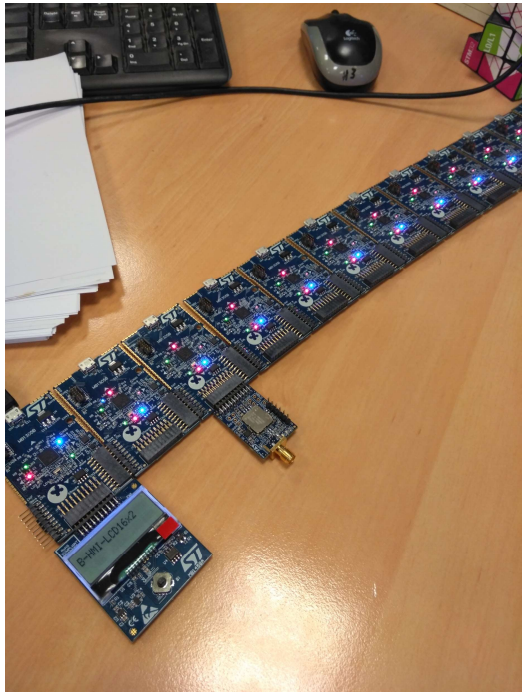
life.augmented

Brisk Quick Gallery History

life.augmented

Brisk

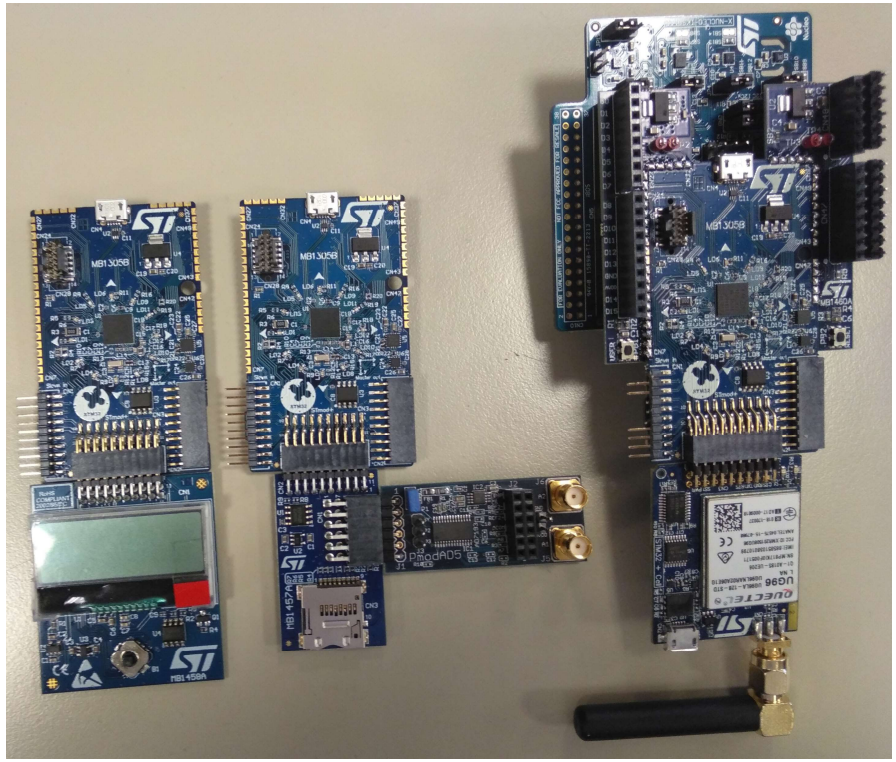
Gen 1: STM32L496

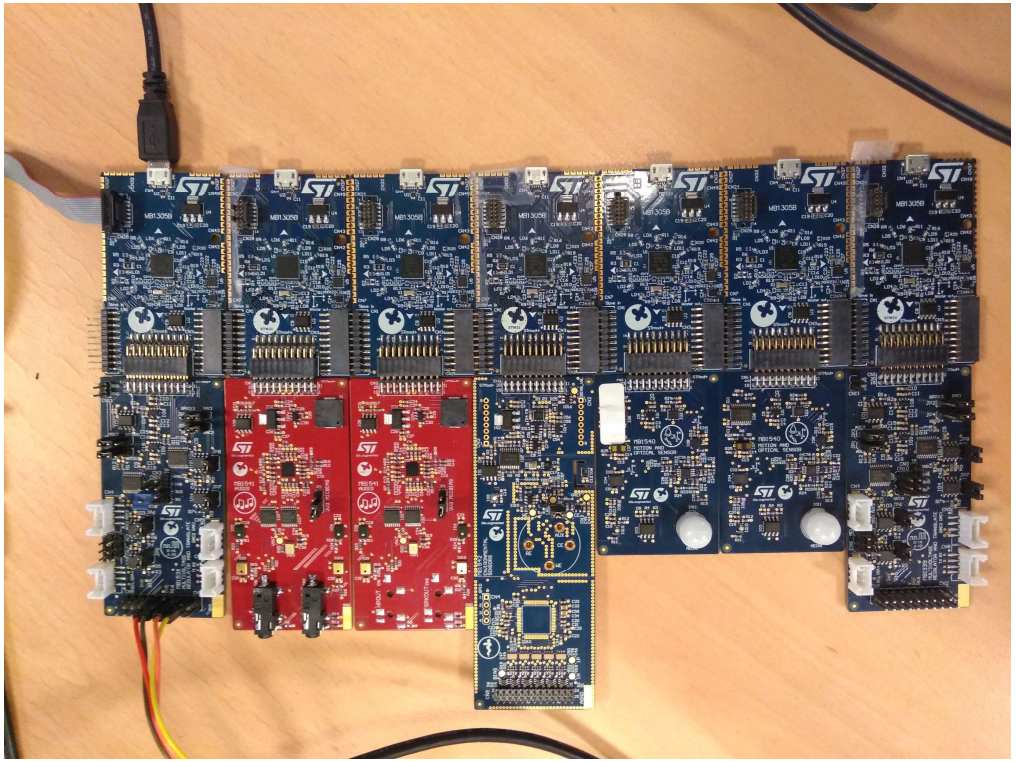


life.augmented

Brisk

Gen 2: STM32L4R

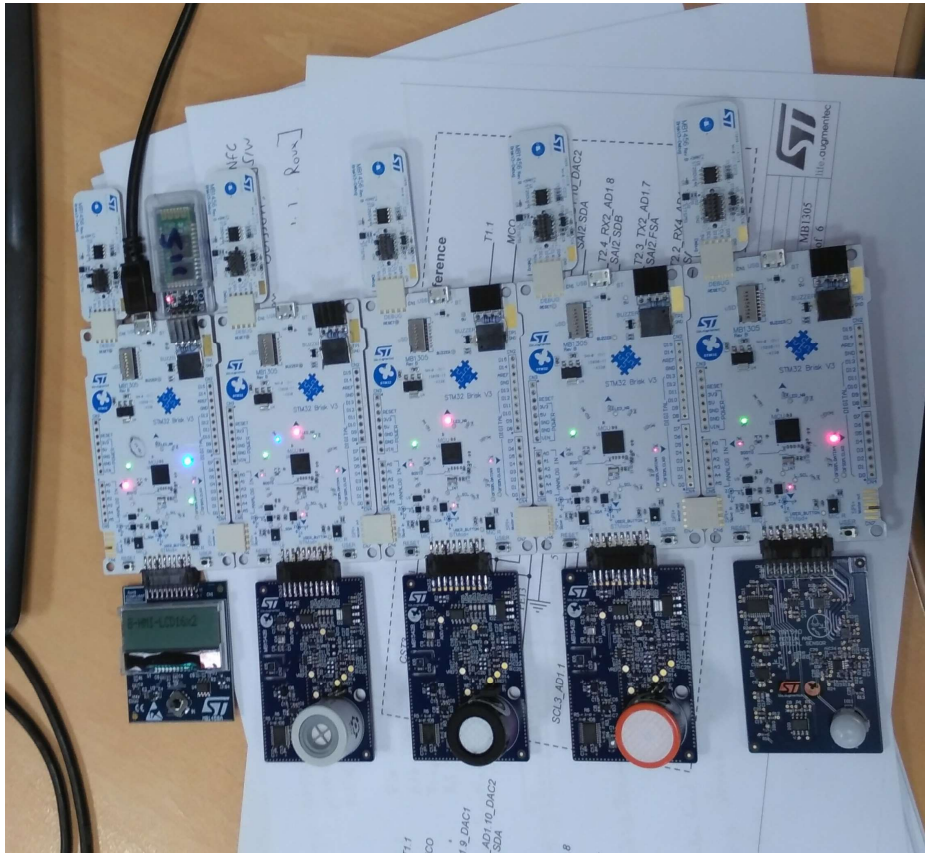


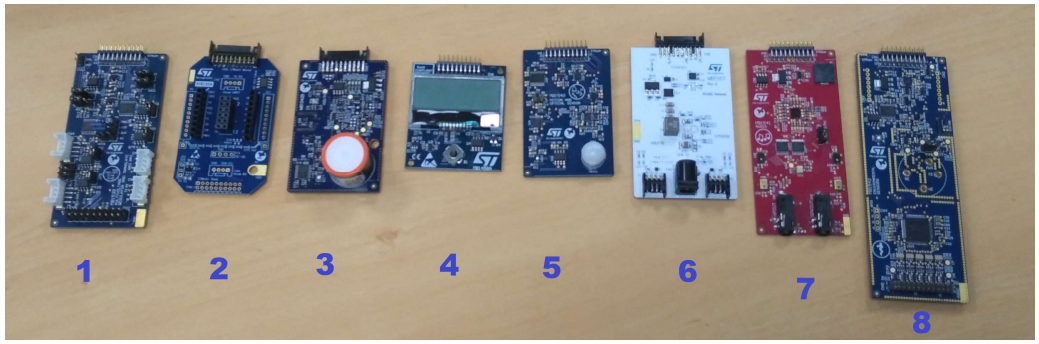


life.augmented

Brisk

Gen 3: STM32L4R

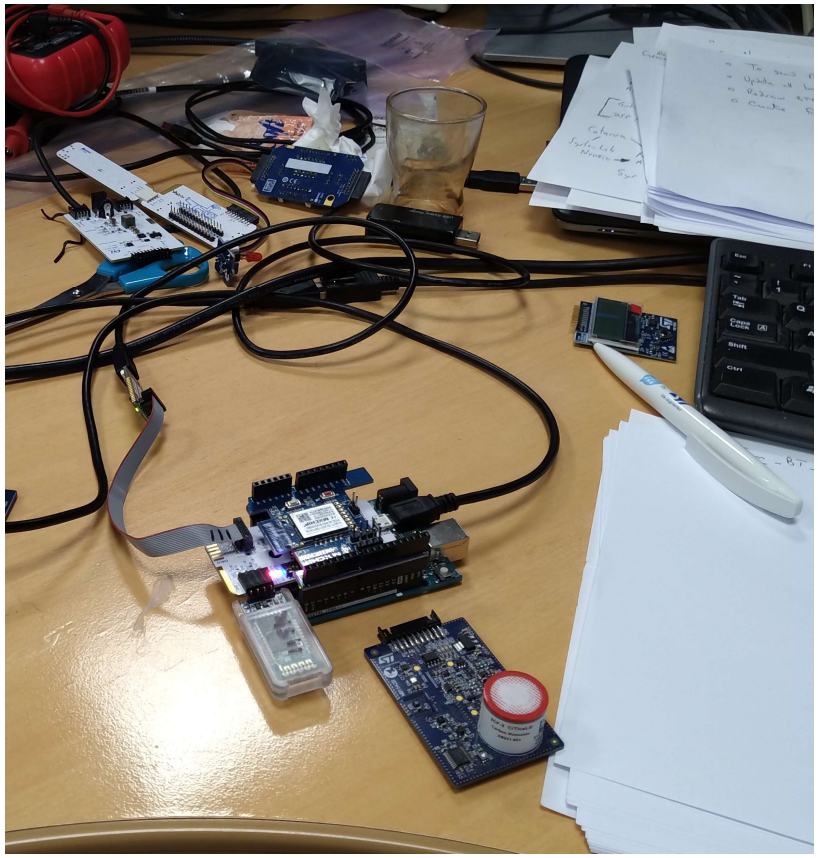


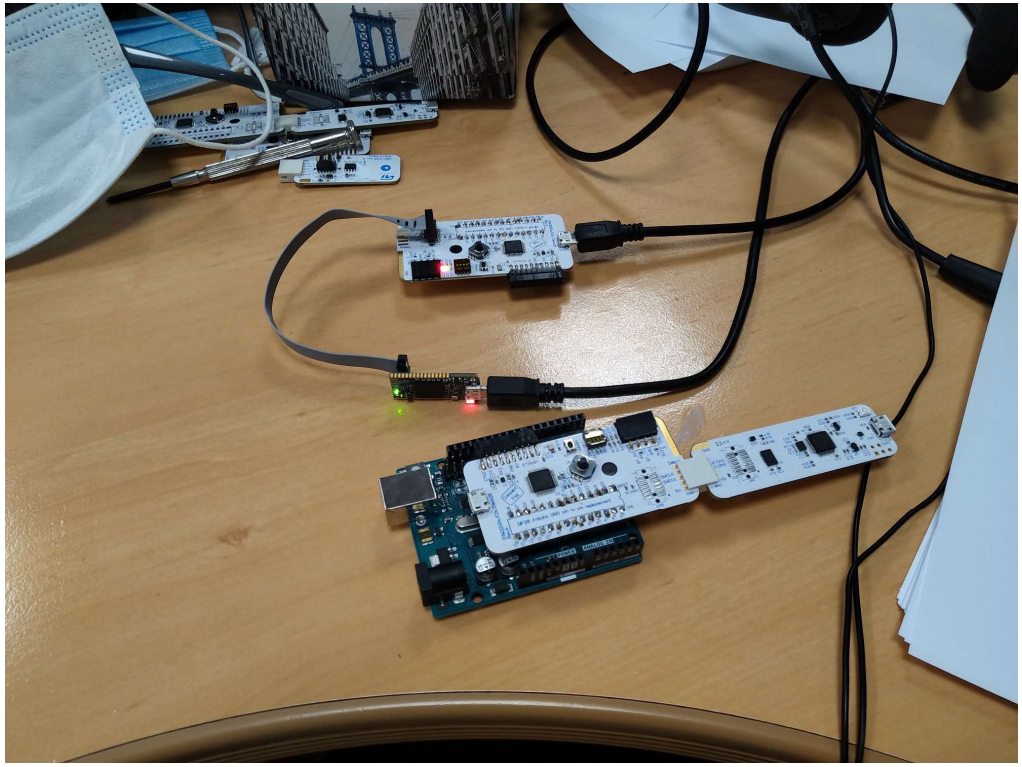


life.augmented

Brisk Subset

Ported over STM32C0316-DK



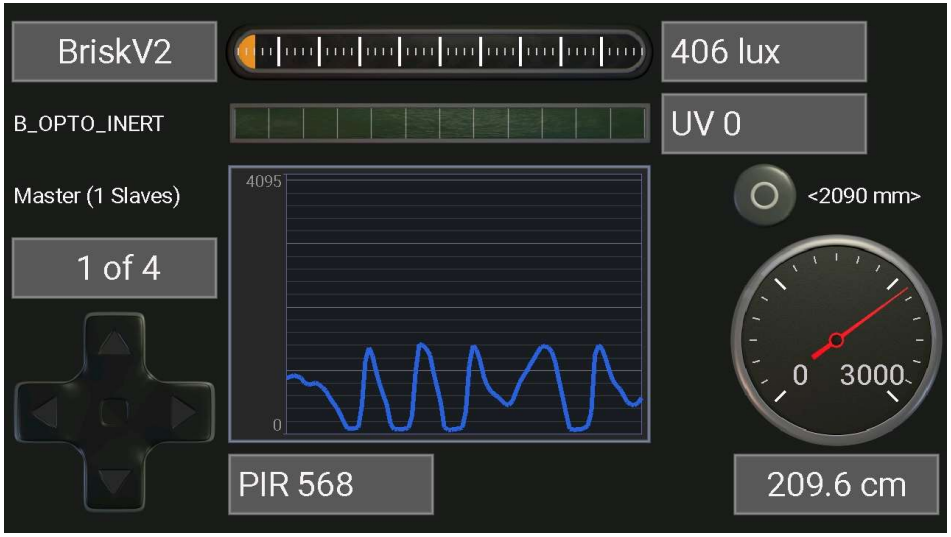


life.augmented

Bluetooth Electronics

Android App

- Cheap RS232 ➔ BT dongle
- Compact RS232 protocol (115200,8,n,1)
- Enable remote dashboard display
- Dashboard layout pushed from MCU to App

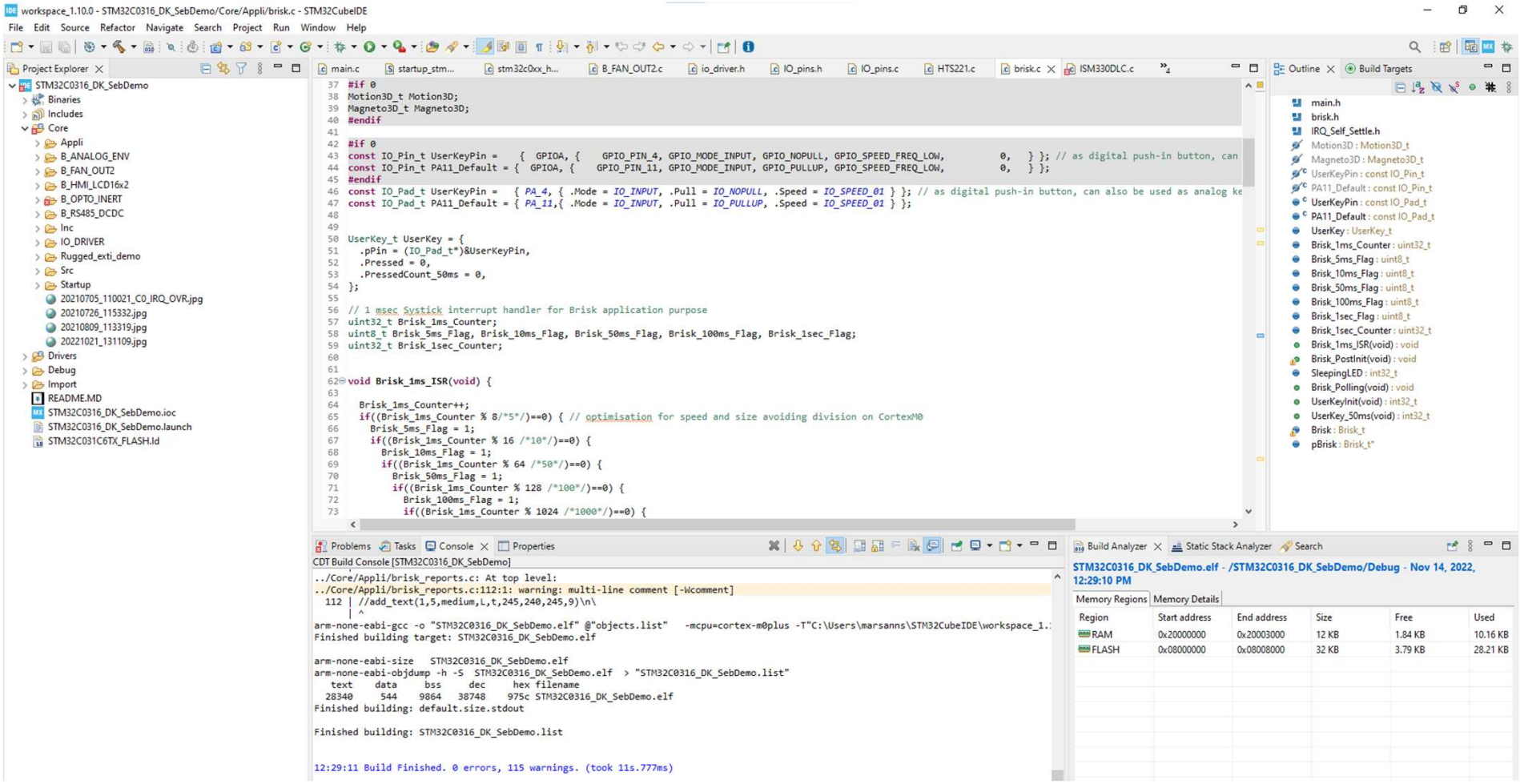


life.augmented

Demo Project quick details

Environment Snapshot

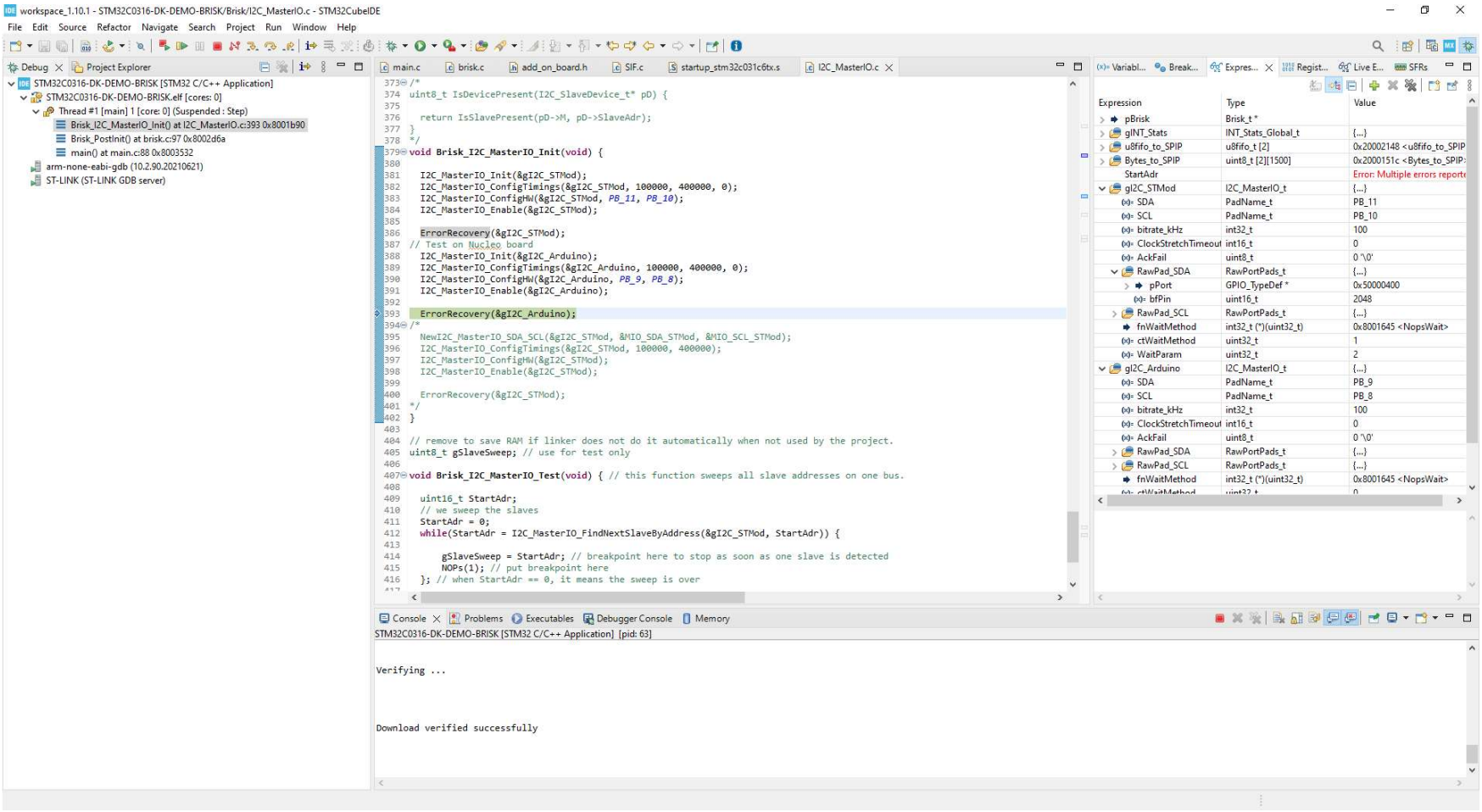
CubeIDE 1.10.0



life.augmented

Environment Snapshot

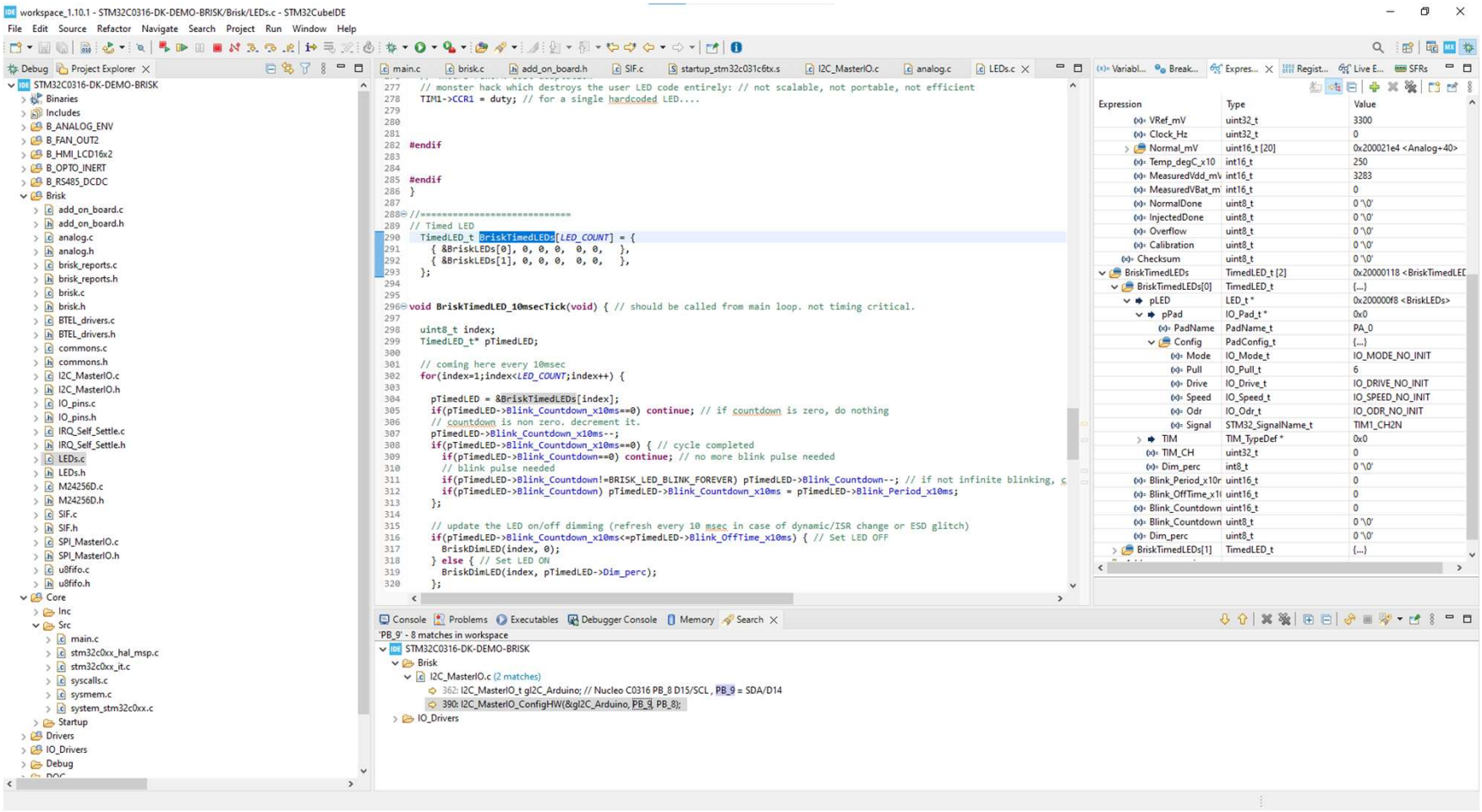
CubeIDE 1.10.0



life.augmented

Environment Snapshot

CubeIDE 1.10.0



life.augmented

Functional blocks

Brief list

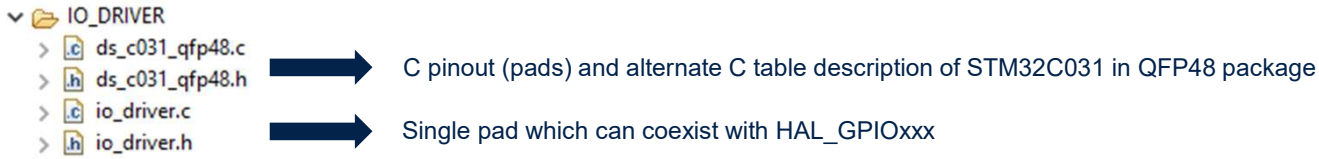
- IO_Driver is an alternate way to control IO Pins more intuitively
- I2C_MasterIO provides I2C Master using IO Bit banging to access Slave Devices (sensors, eeprom)
- SPI_MasterIO provides SPI Master using IO Bit banging to drive a 2x16 characters-based LCD
- LEDs enable list of LEDs with digital blinking and/or dimming (PWM as %)
- U8FIFO is a SW Fifo to plumb data between asynchroneous data provider and consumer
- SIF and BTEL_Drivers are serial data using Bluetooth Electronics protocol to use remote Phone display
- The STMod+ Add-on boards are plug and play and contain various sensors and other things
  - Add_on_board.c perform the plug and play by regularly searching the presence of the add-on board EEPROM ID
  - Each add-on board is named “B_xxxxx” and its source files and documents are placed in a sub folder
  - Due to small STM32C03 memory, only 1 or 2 boards can be supported with limited sensor support (no float) ➔ add_on_board.h
- Brisk is the top application level called from main()
- Analog perform one shot multiple channels including measuring STM32 Vdda and Temperature

SW Elements

IO_Driver

GPIOs can be controlled by either HAL and LL.
*These APIs require a pin, defined by a **set** (GPIO port, BitMask).*

*IO_Driver focuses on intuitiveness and easy to use APIs using a **single** pin name.*
*IO_Driver uses "**Pads**" to differentiate and coexist with HAL and LL.*



Browse the application code to discover its use, which was previously using HAL.
The IO_Driver check pins validity, is interrupt proofed, deals with EXTI and makes pin configuration friendlier with an overall efficient generated code. For bit-bang speedy Ios, RawPads are introduced to make I2C_Master or SPI_Master emulation.

Datasheet package pinout and alternate functions signals have been C encoded too.

life.augmented

SW Elements

IO_Driver

```
Search for these API in the application code to find out how it is used… Check the MAP file for code density, check I2C_Master_IO for I2C Bitbang on any pin
If you'd like to migrate from HAL to IO_Driver step by step, use IO_Pins.c/h which first create Pad APIs using HAL, then rename pin into pad to finish the conversion….

int32_t IO_PadInit(IO_Pad_t  *pIO_Pad);// intuitive
int32_t IO_PadDeInit(PadName_t PadName); // intuitive
int32_t IO_PadGet(PadName_t PadName); // thread safe, speed
int32_t IO_PadSetHigh(PadName_t PadName); // thread safe, speedy
int32_t IO_PadSetLow(PadName_t PadName); // thread safe, speedy
int32_t IO_PadSet(PadName_t PadName, GPIO_PinState PinState); // thread safe, speedy
int32_t IO_PadToggle(PadName_t PadName); // thread safe, speedy
HAL_StatusTypeDef IO_PadLock(PadName_t PadName);
//void HAL_IO_EXTI_IRQHandler(PadName_t PadName); // moved to EXTI driver
//void HAL_IO_EXTI_Rising_Callback(PadName_t PadName);
//void HAL_IO_EXTI_Falling_Callback(PadName_t PadName);


void IO_RawPadSetHigh(RawPortPads_t* RawPad);
void IO_RawPadSetLow(RawPortPads_t* RawPad);
uint32_t IO_RawPadGet(RawPortPads_t* RawPad);

int32_t IO_EXTI_Config(PadName_t PadName, EXTI_Config_t EXTI_Config);
int32_t IO_EXTI_DeInit(PadName_t PadName);

//  IO_Pin_t MIO_SDA_STMod = {  GPIOB,          {    GPIO_PIN_11, GPIO_MODE_OUTPUT_OD, GPIO_PULLUP, GPIO_SPEED_FREQ_LOW, 0,   } };
IO_Pad_t SDA = { PB_11, {.Mode = IO_OUTPUT, .Pull = IO_PULLUP, .Drive = IO_OPENDRAIN, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH} };

const IO_Pad_t LED1_AsOutput = {  PA_5, { .Mode = IO_OUTPUT, .Drive = IO_PUSHPULL, .Pull = IO_NOPULL, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH } };
// LED1 = Digital Output, the indefined configuration fields will remain unchanged. .Odr field enables set the output level before turning the pin as output.

const IO_Pad_t LED1_AsPWM =    {  PA_5, { .Mode = IO_SIGNAL, .Drive = IO_PUSHPULL, .Pull = IO_NOPULL, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH, .Signal = TIM1_CH1/*5*/ } };
// LED1 = Digital PWM T1.1 AF5 IO_SIGNAL means use alternate function signal. No number needed (portability), only valid signal name: TIM1_CH1
```

life.augmented

SW Elements

I2C_MasterIO overview

I2C_MasterIO is an I2C Master implementation with just GPIO and SW. Useful for board bringup and flexible pin assignment, more core bandwidth intensive than using a HW I2C.

The driver defines the I2C bus properties, pins, speed, and slave configuration.
By default, this code does not support clock stretching and drive true slaves only.

int32_t I2C_MasterIO_AccessSlave(I2C_SlaveDevice_t pD);*
The only API needed to drive all the I2C Slave supported in the Application (browse it)
APIs are provided to sweep of all I2C Slave addresses, handy for board bring up.
The module has a local test function as example.

Slave code example are for example M24256D (EEPROM), STTS751, BME280, LPS22HH….

Note: Sensirion sensor is not a true I2C Slave and has its own bit bang driver.

life.augmented

SW Elements

I2C_MasterIO snapshot

```
M24256D API extract: 2 bytes sub address for 32 kbyte data with pages of 64 bytes. Here is the API to write data in ID OTP Page and verify.
uint8_t EEP_WriteID (EEP_t* pEEP) {

  //uint8_t RetryCount = MAX_RETRIALS_ON_ERROR;
  uint8_t i,buf[EEPPAGESIZE_BYTE];
  uint8_t result = 1;
  I2C_SlaveDevice_t* pDevice = pEEP->pDeviceID;

  pDevice->SubAdrBytes[0] = 0x00;// msb
  pDevice->SubAdrBytes[1] = 0x00;// lsb;
  pDevice->pWriteByte = pEEP->ID_Page;
  pDevice->WriteByteCount = EEPPAGESIZE_BYTE; // no data write
  pDevice->pReadByte = 0;
  pDevice->ReadByteCount = 0;
  result =  I2C_MasterIO_AccessSlave(pDevice);
  //
  EEP_WaitWriteCompletion(pEEP); // to mature
  for(i=0;i<EEPPAGESIZE_BYTE;i++) // backup original
    buf[i] = pEEP->ID_Page[i];

  // verify
  EEP_ReadID (pEEP);
  for(i=0;i<EEPPAGESIZE_BYTE;i++) {
    if(buf[i] != pEEP->ID_Page[i])
      break;
    pEEP->ID_Page[i] = buf[i];
  };
  if(i != EEPPAGESIZE_BYTE)
    TrapError(); // warning for hot plug (not supposed to happen for ID Page)

  return result; // if 0 : writing failed
```

life.augmented

...

- To be continued based on feedback

life.augmented

Project Source License Inventory

life.augmented

Project author’s source code

file header

```
/******************************************************************************
  * @file    add_on_board.c (STMod+ plug and play add-on board manager)
  * @author  S.Ma
  * @brief   (…)
  *
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2018-2022 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
```

life.augmented

CMSIS

license.txt

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

http://www.apache.org/licenses/LICENSE-2.0

life.augmented

HAL/LL

file header

ST Sensor Driver License

Motion Mems and Environmental

life.augmented

ST Sensor Driver License

Optical Time Of Flight

life.augmented

Bosch Sensor

Header

/** \mainpage **

* Copyright (C) 2010 - 2015 Bosch Sensortec GmbH

* File : bmg160.h

* Date : 2015/04/29

* Revision : 2.0.4 $

* Usage: Sensor Driver for BMG160 sensor

****

* \section License

**/

Our technology starts with You

Find out more at www.st.com

life.augmented

Bonus

- The project has a “Rugged EXTI demo” (IRQ_Self_Settle source files)
- Simple demo of how to prevent EXTI IO interrupt “DDOS” by hacker injecting MHz square wave onto an IO Interrupt enabled pin
- The demo uses the Fan Out board (picture provided)
  - The MCO pin is enabled to output 8 MHz from STM32
  - One of the STMod+ pin is configured as EXTI interrupt
  - Using a jumper wire to short MCO to EXTI will flood the code with interrupts and hang it
    - The green beat pulsing LED will freeze
  - Using the user rock switch, the countermeasure can be enabled so the code become immune
    - A Timer delay is used to delay the EXTI interrupt reactivation limiting the CPU interrupt load

For further support in creating a PowerPoint presentation, including graphic assets, formatting tools and additional information on the ST brand

you can visit the ST Brand Portal

https://brandportal.st.com

life.augmented