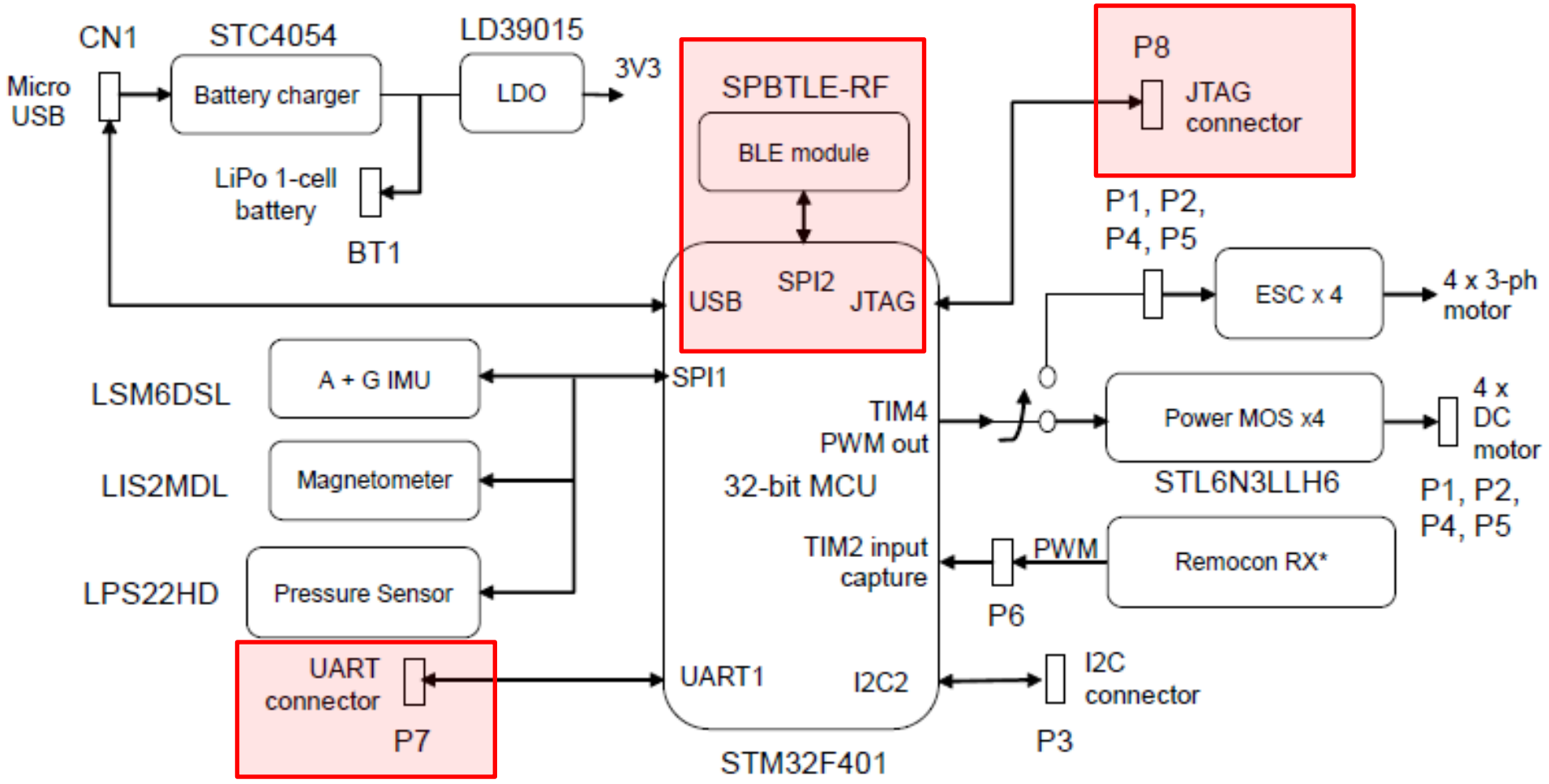
Wireless Communication & Remote Control

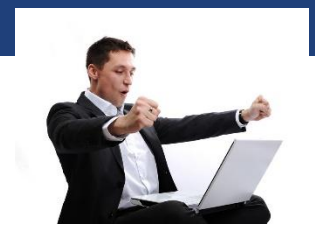
Project-based Learning Center | ETH Zurich
Tommaso Polonelli tommaso.polonelli@pbl.ee.ethz.ch
Michele Magno michele.magno@pbl.ee.ethz.ch
Vlad Niculescu vladn@iis.ee.ethz.ch

Bluetooth Low Energy
&
Sensors

HW overview

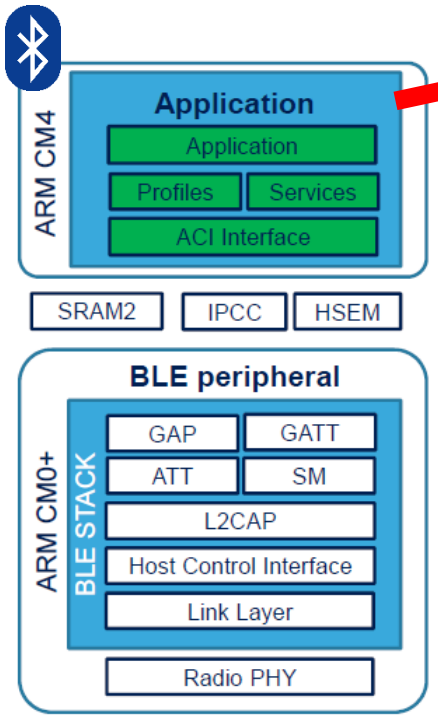

CN1
Micro USB
STC4054
Battery charger
LD39015
LDO
3V3
LiPo 1-cell battery
BT1
SPBTLE-RF
BLE module
USB
SPI2
JTAG
P8
JTAG connector
P1, P2, P4, P5
ESC x 4
4 x 3-ph motor
LSM6DSL
A + G IMU
SPI1
LIS2MDL
Magnetometer
LPS22HD
Pressure Sensor
TIM4 PWM out
Power MOS x4
STL6N3LLH6
4 x DC motor
P1, P2, P4, P5
32-bit MCU
TIM2 input capture
PWM
Remocon RX*
P6
UART connector
P7
UART1
I2C2
I2C connector
P3
STM32F401

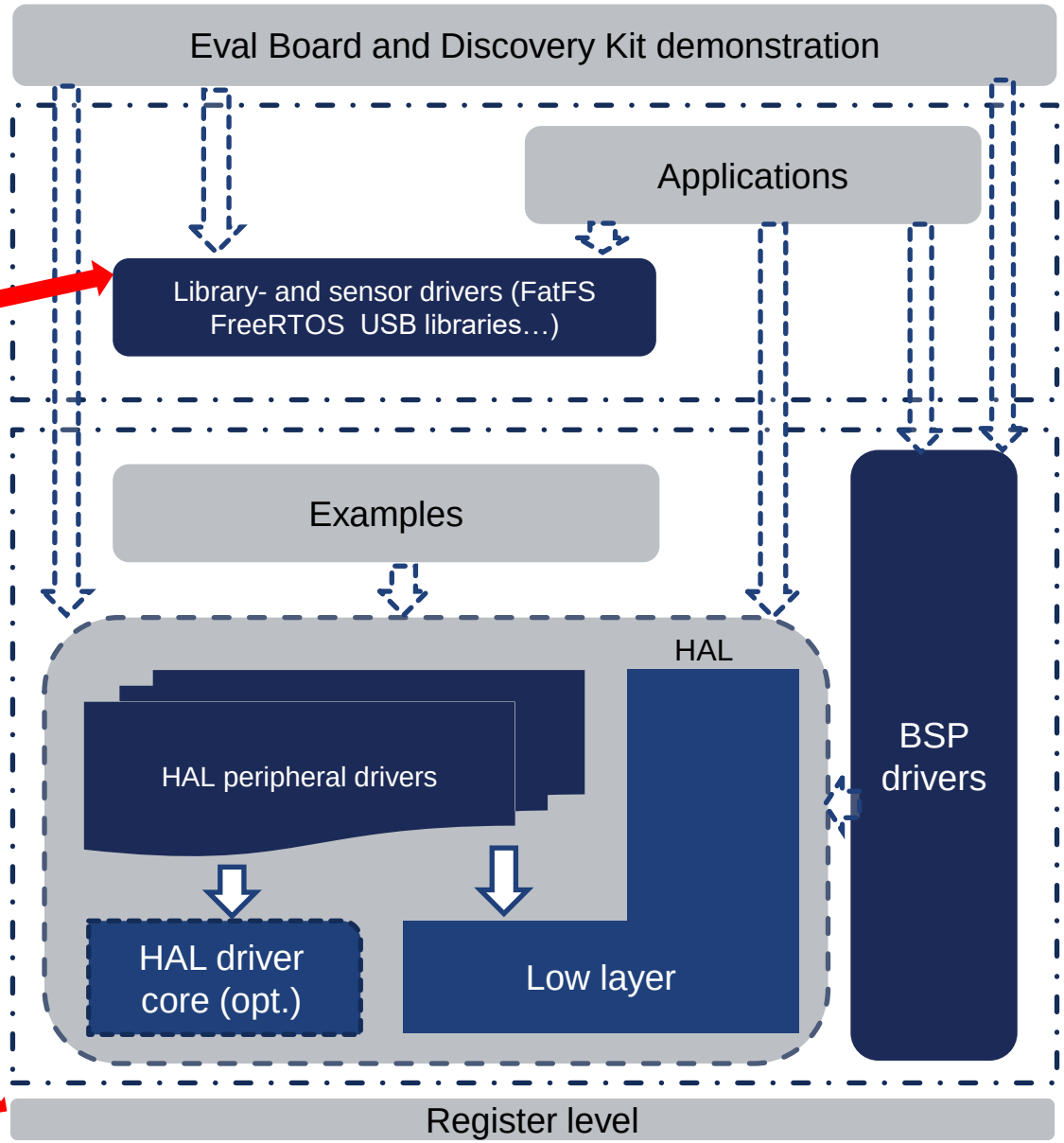
STM32 software architecture



YOU

Application Level 2





STWIN Template

Advanced

BLE Stack

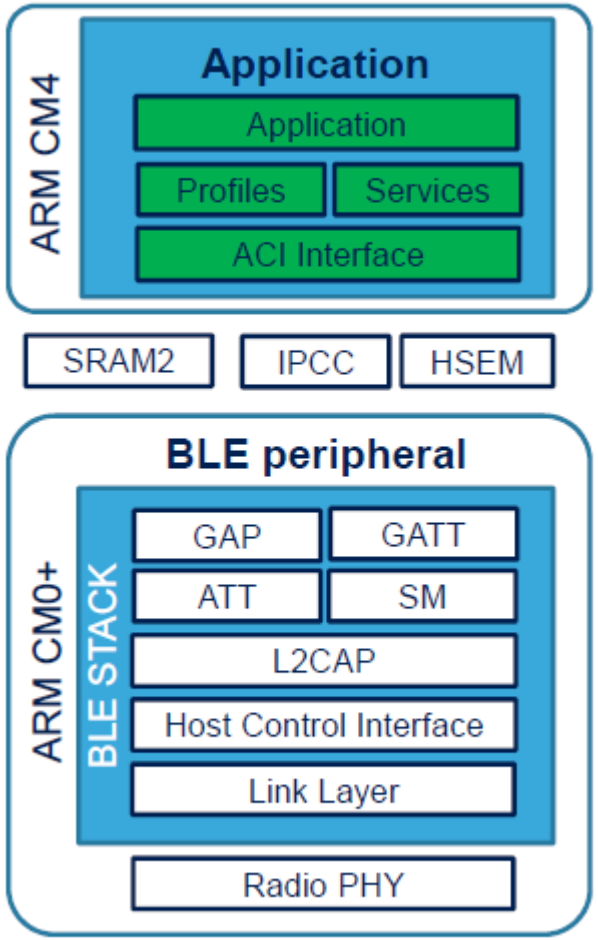
stm32l4xx_it.c | main.c

```
758   /* get the BlueNRG
```

STM32L4

mailbox system

BLE Module



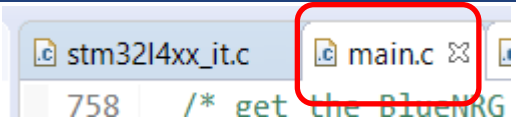

```
/* Initialize the BlueNRG Custom services */
Init_BlueNRG_Custom_Services();
```

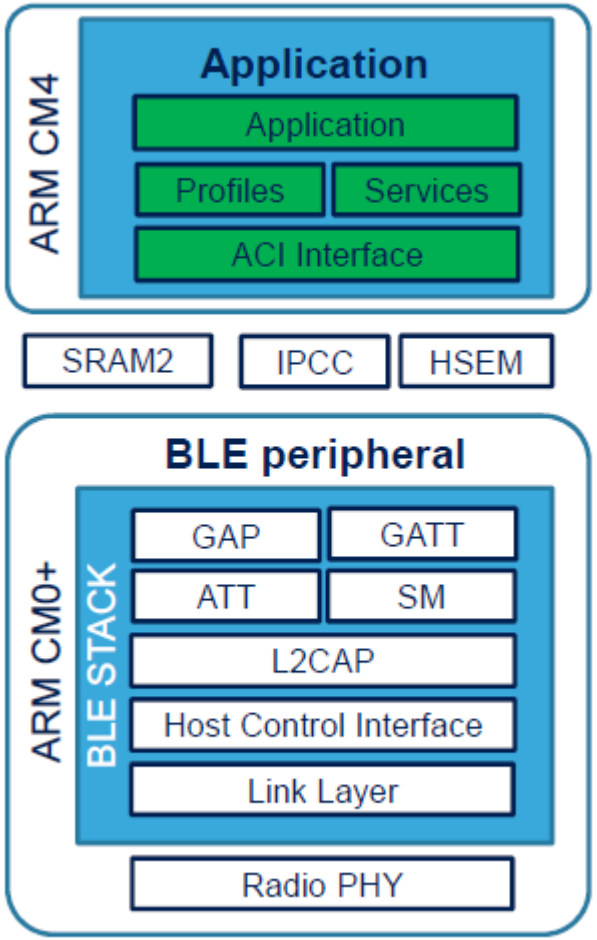
```
BNRG_SPI_Init();
```

```
/* Initialize the BlueNRG */
HCI_Init();
```

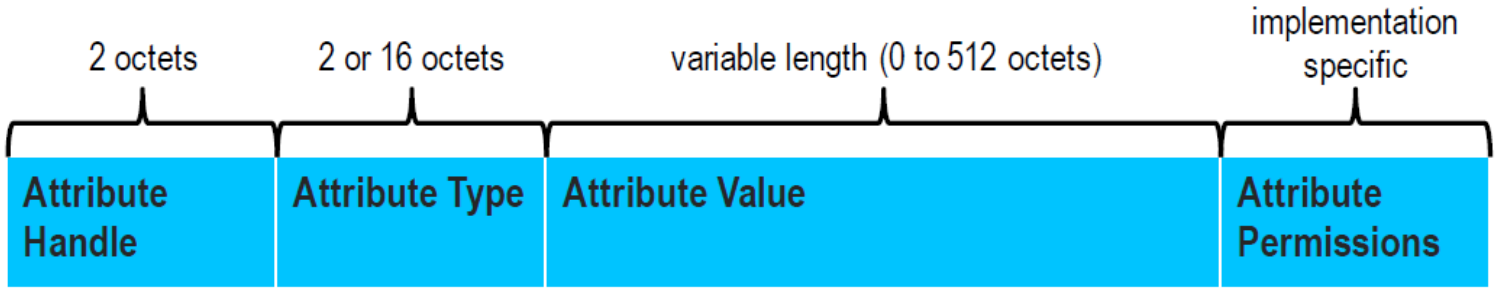
```
BlueNRG_Init();
```

BLE Stack: Application and Services






```
/* Initialize the BlueNRG Custom services */
  Init_BlueNRG_Custom_Services();
```



Attributes:

- Type: UUID → determines what the value means
- Types are defined by “Characteristic Specification” or Generic Access Profile or Generic Attribute Profile

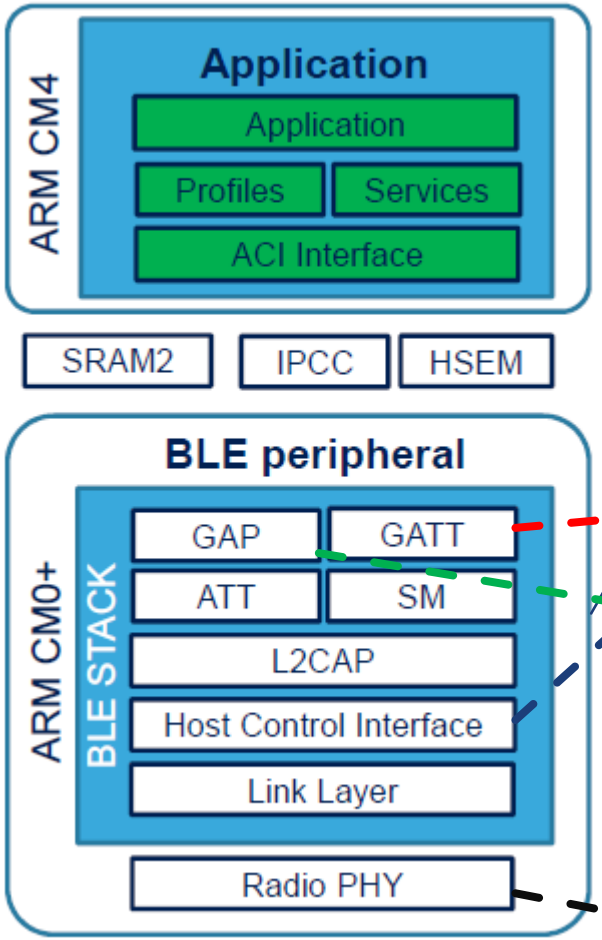
```
/* Hardware & Software Characteristics Service */
ENVIRONMENTAL_UUID UUID_128(00 00 00 00 00 01 11 e1 ac 36 00 02 a5 d5 c5 1b)
ACC_GYRO_MAG_UUID  UUID_128(00 E0 00 00 00 01 11 e1 ac 36 00 02 a5 d5 c5 1b)
MIC_UUID           UUID_128(04 00 00 00 00 01 11 e1 ac 36 00 02 a5 d5 c5 1b)
BATTERY_UUID       UUID_128(00 02 00 00 00 01 11 e1 ac 36 00 02 a5 d5 c5 1b)
```

`uuid_ble_service.h`

BLE Stack: GATT, GAP, HCI, PHY

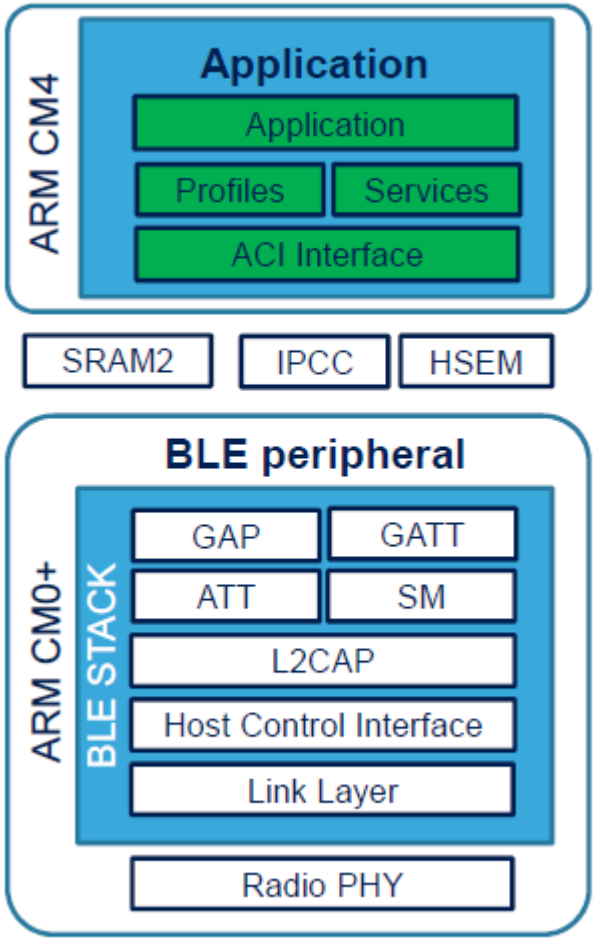
stm32l4xx_it.c | main.c

758 /* get the BlueNRG




```
/* Initialize the BlueNRG */
  Init_BlueNRG_Stack();

/* Initialize the BlueNRG SPI driver */
 hci_init(APP_UserEvtRx, NULL);

 ret = aci_gatt_init();

 ret = aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x07,
 &service_handle, &dev_name_char_handle,
 &appearance_char_handle);

 /* Set output power level: -2,1 dBm */
  aci_hal_set_tx_power_level(1,4);
```

Update Sensor Data




```
/**
  * @brief  Send Audio Level Data (Ch1) to BLE
  * @param  None
  * @retval None
  */
static void SendAudioLevelData(void)
{
  int32_t NumberMic;
  uint16_t DBNOISE_Value_Ch[AUDIO_IN_CHANNELS];

…
…
…

  AudioLevel_Update(DBNOISE_Value_Ch);
}
```

stm32l4xx_it.c | main.c

```
758    /* get the BlueNRG
```

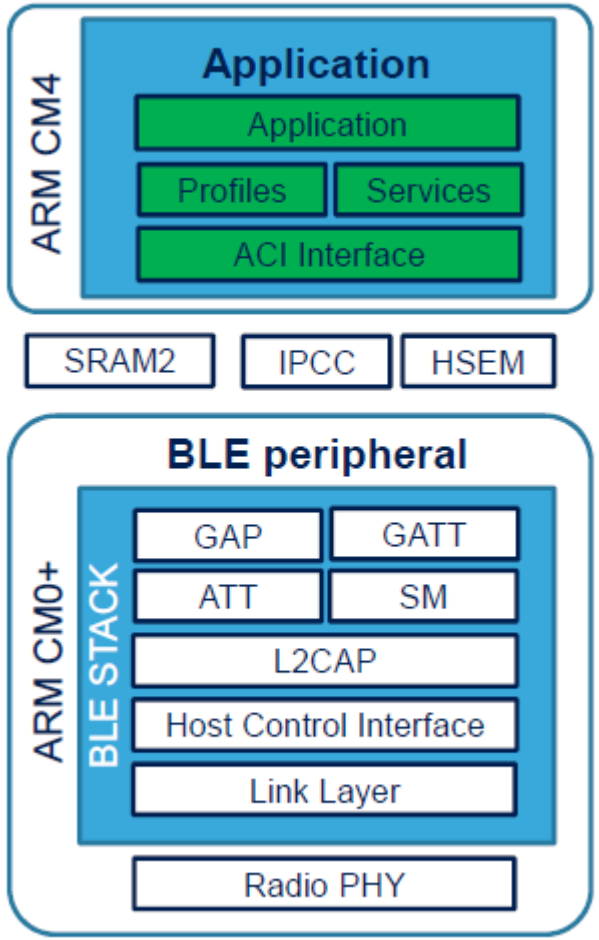
Update the attribute value of a specific attribute type (UUID)

aci_gatt_update_char_value

stm32l4xx_it.c | main.c | MotionSP_Manager.c | sensor_service.c

```
705    }
```

Remote control characteristics



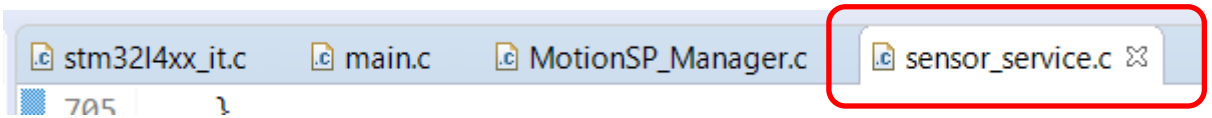
Receive commands

```
  /* MAX charecteristic */
  COPY_MAX_W2ST_CHAR_UUID(uuid);
ret =  aci_gatt_add_char(HWServW2STHandle, UUID_TYPE_128, uuid, 7,
                         CHAR_PROP_WRITE_WITHOUT_RESP | CHAR_PROP_WRITE,
                         ATTR_PERMISSION_NONE,
                         GATT_NOTIFY_ATTRIBUTE_WRITE,
                         16, 0, &MaxCharHandle);
```

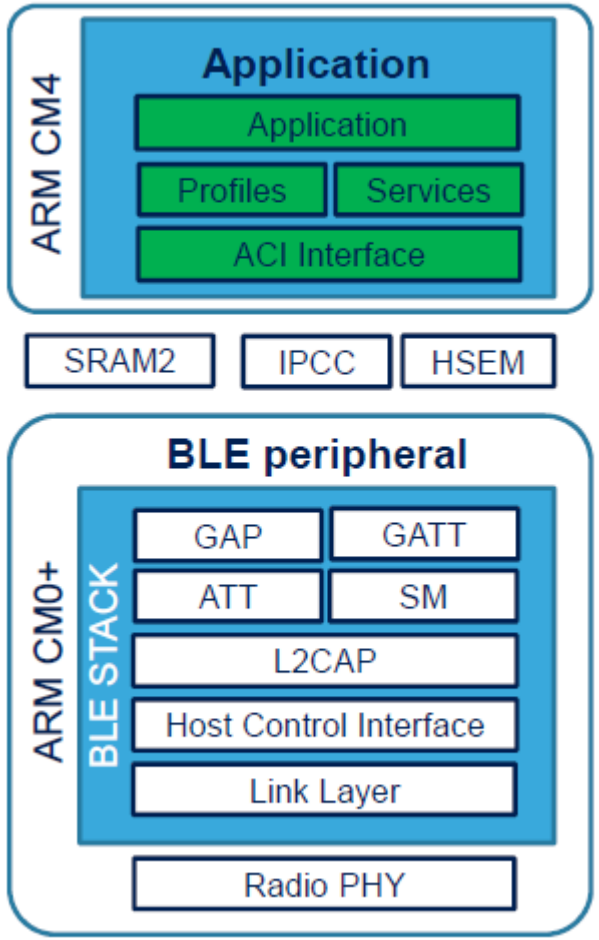
Update the attribute value of a specific attribute type (UUID)

Used to poll (read request from RC) drone state

```
COPY_ARMING_W2ST_CHAR_UUID(uuid);
ret =  aci_gatt_add_char(HWServW2STHandle, UUID_TYPE_128, uuid, 2+1,
                         CHAR_PROP_NOTIFY | CHAR_PROP_READ,
                         ATTR_PERMISSION_NONE,
                         GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP,
                         16, 0, &ArmingCharHandle);
```



Remote control characteristics



Receive commands

```
  /* MAX charecteristic */
  COPY_MAX_W2ST_CHAR_UUID(uuid);
ret =  aci_gatt_add_char(HWServW2STHandle, UUID_TYPE_128, uuid, 7,
                         CHAR_PROP_WRITE_WITHOUT_RESP | CHAR_PROP_WRITE,
                          ATTR_PERMISSION_NONE,
                          GATT_NOTIFY_ATTRIBUTE_WRITE,
                          16, 0, &MaxCharHandle);
```

```
joydata[0]
joydata[1]
joydata[2]
joydata[3]
joydata[4]
joydata[5]
joydata[6]
joydata[7]
```

Callback for every WRITE operation

From APP to Drone

```
void Attribute_Modified_CB(uint16_t
attr_handle, uint8_t * att_data, uint8_t
data_length)
```

Remote control data format (main.c – main – while(1)

Receive commands

```
//          gTHR = joydata[4]*13;
//          gAIL = (joydata[5]-128)*(-13);
//          gELE = (joydata[6]-128)*13;
gRUD = (joydata[2]-128)*(-13);
gTHR = joydata[3]*13;
gAIL = (joydata[4]-128)*(-13);
gELE = (joydata[5]-128)*13;
```

Data conversion

```
/* joydata[6]: seek bar data*/
/* joydata[7]: additional button data
           first bit: Takeoff (0 = Land,  1 = Takeoff)
           second bit: Calibration When it changes status is active
           third bit: Arming (0 = Disarmed,  1 = Armed) */
gJoystick_status = joydata[7];
if ((gJoystick_status&0x04)==0x04){
    rc_enable_motor = 1;
    fly_ready = 1;
    BSP_LED_On(LED2);
}
else {
    rc_enable_motor = 0;
    fly_ready = 0;
}
```

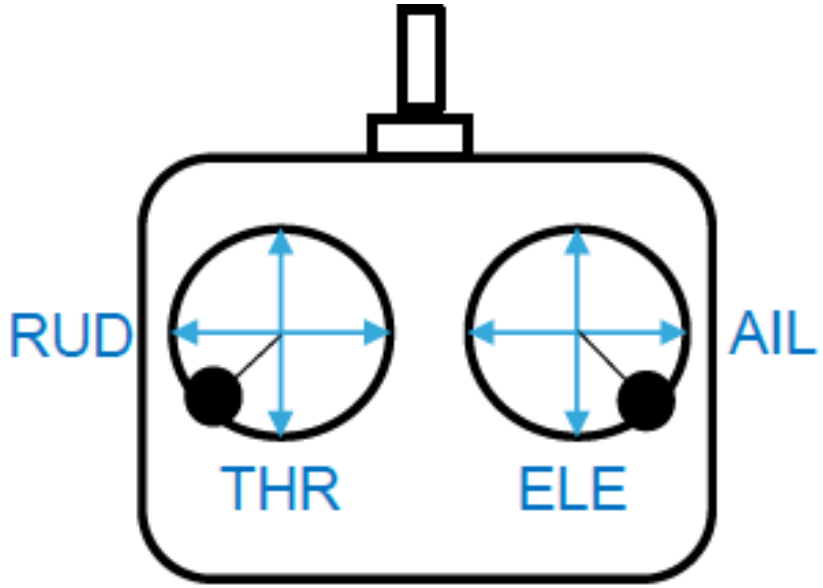
Arming

gTHR = throttle

gAIL = roll

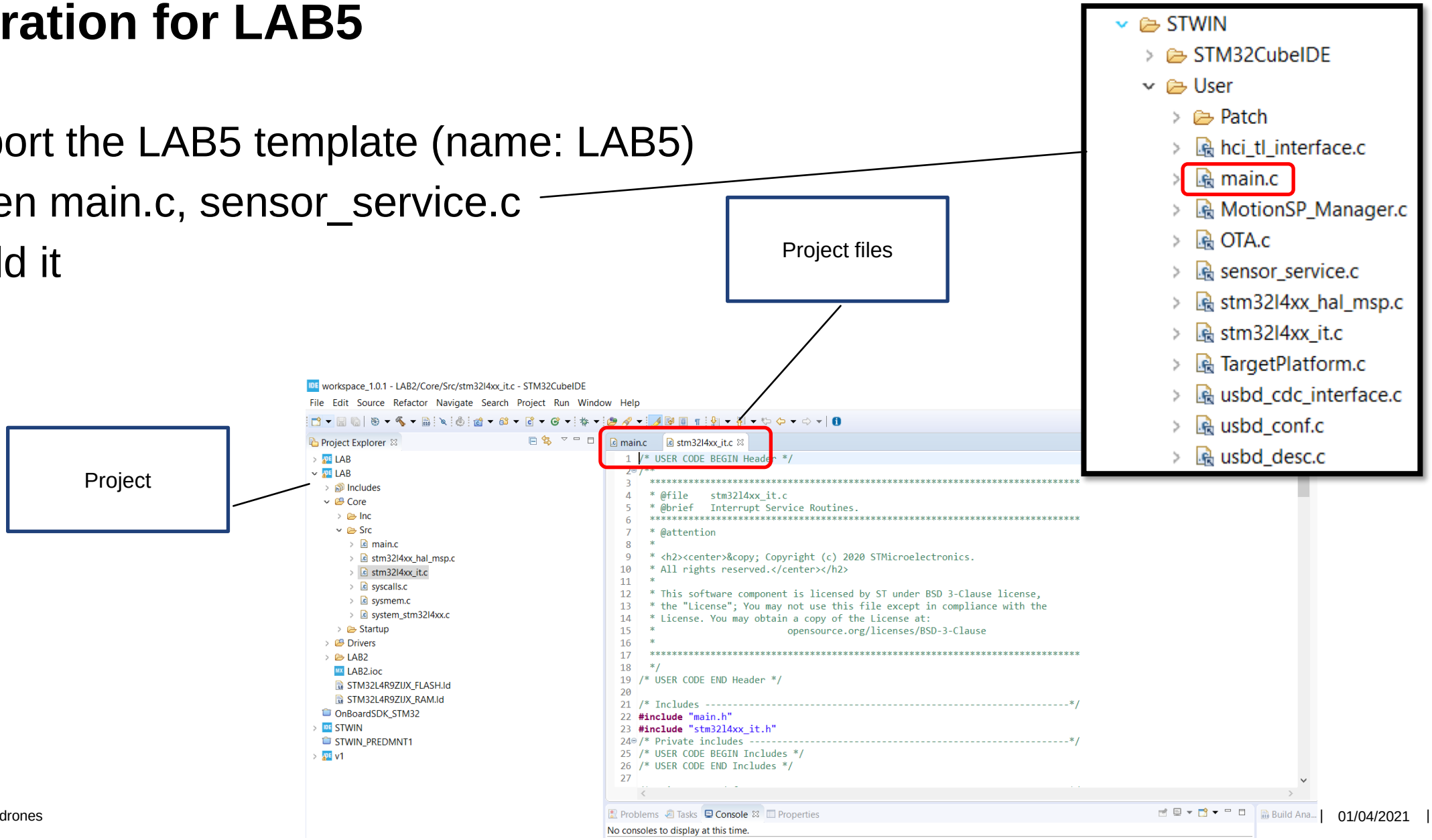
gELE = pitch

gRUD = yaw

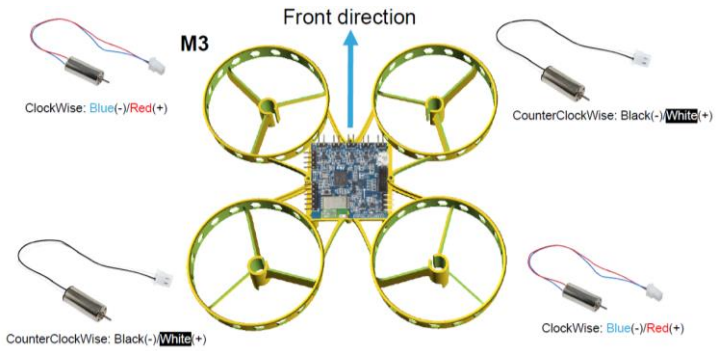


Preparation for LAB5

- Import the LAB5 template (name: LAB5)
- Open main.c, sensor_service.c
- Build it



STWIN application tools



UART

serial communication

ST BLE Drone

Tutorial: https://www.youtube.com/watch?v=jxznkY2QUJ8

App: https://www.st.com/en/embedded-software/st-ble-drone.html

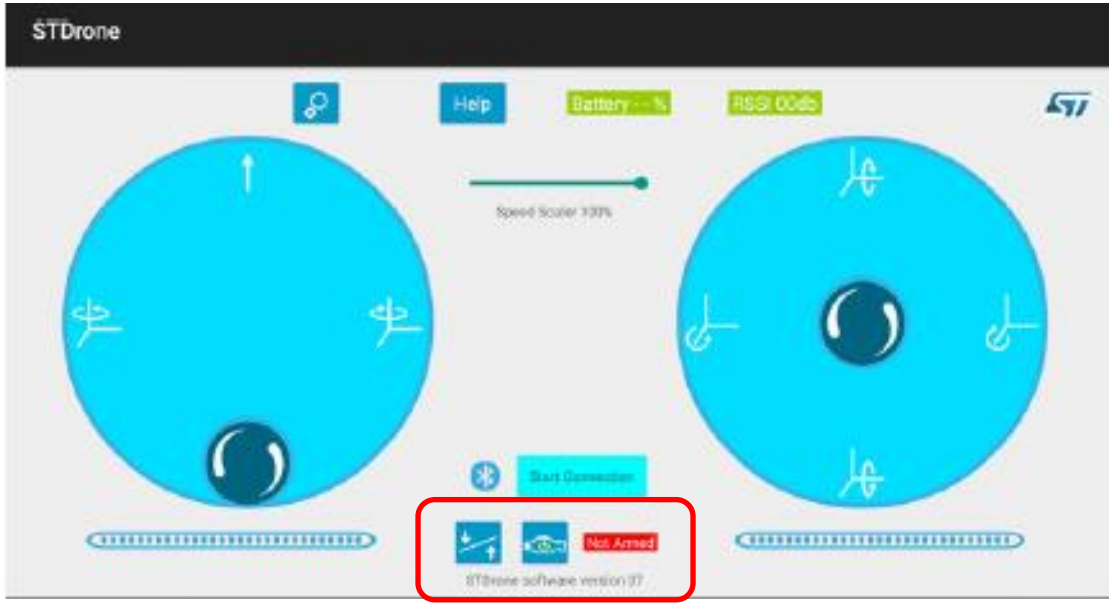
Source code on polybox!

Arming procedure

Step1: Place the quadcopter on a flat surface and press the reset button on the board.

Step2: Click [Start Connection] in the main app screen.

Step3: In the AppDrone app, tap on the icon. 



Test flight

Step1: Slide the speed scaler down to 60%.

Step2: Select the settings icon and set the left joystick capability to throttle only.

Step3: In the AppDrone app, tap on the icon. 

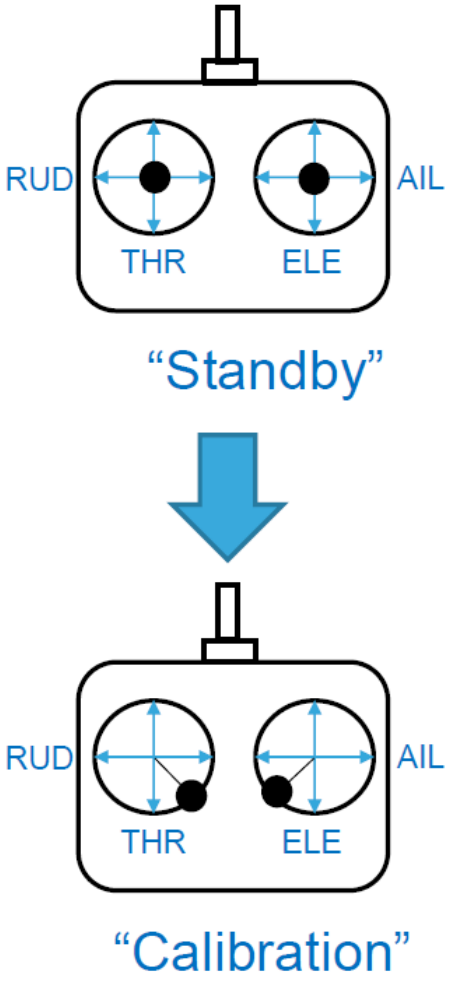
LAB5: Exercise overview

Template can be found in your Polybox folder LAB5

Exercise	Assignment	Concept
Exercise 1	Import and execute the template. Connect the sensor node with your smartphone and plot on the terminal gTHR, gAIL, gELE, gRUD	USB BLE
Exercise 2	Only when the drone is armed: modify the motor speed proportionally to gTHR, gAIL, gELE	USB BLE
Exercise 3	Only when the drone is NOT armed: custom sensor calibration request	USB BLE
Exercise 4	Safety check: when the drone is flipped, switch off the motor and disarm the control. The APP has to receive the arming update	USB BLE EXTI
Exercise 5	Finalize LAB4	AHRS

Sensor calibration procedure

Exercise 3



If the battery is applied to the FCU (or reset button pushed) when the quadcopter is not on a flat surface (or the FCU not mounted flat on the frame), the AHRS Euler angle will have an offset. To eliminate it, run a simple calibration procedure through the remote control after start-up.

```
if ( (gTHR == 0) && (gELE < -RC_CAL_THRESHOLD) && (gAIL > RC_CAL_THRESHOLD) && (gRUD < -RC_CAL_THRESHOLD))
  {
    rc_cal_flag = 1;
  }
```