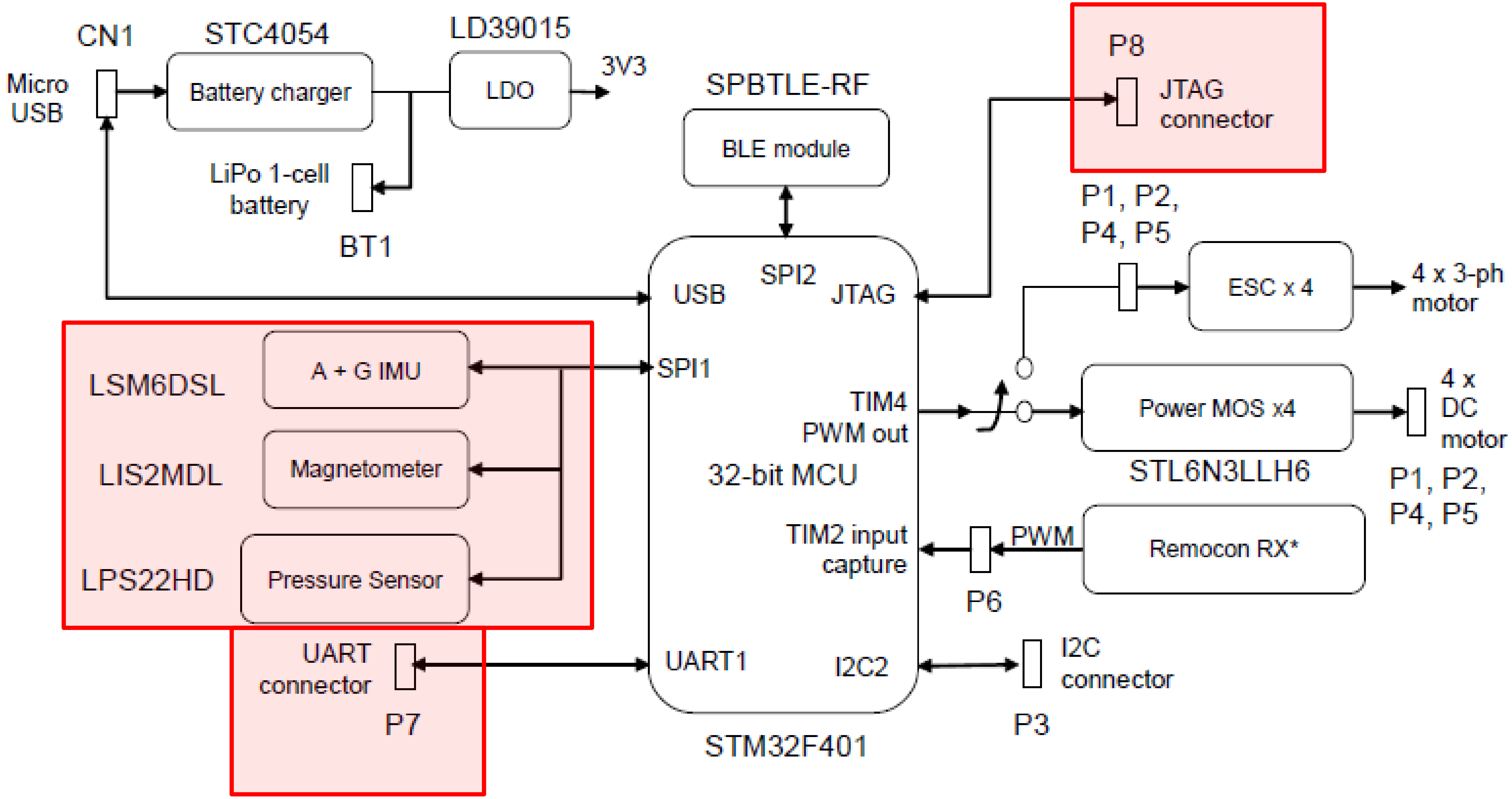
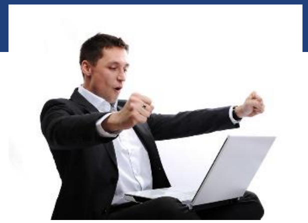
Motion sensors: Pitch, Roll, Yaw

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

HW overview


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

STM32 software architecture



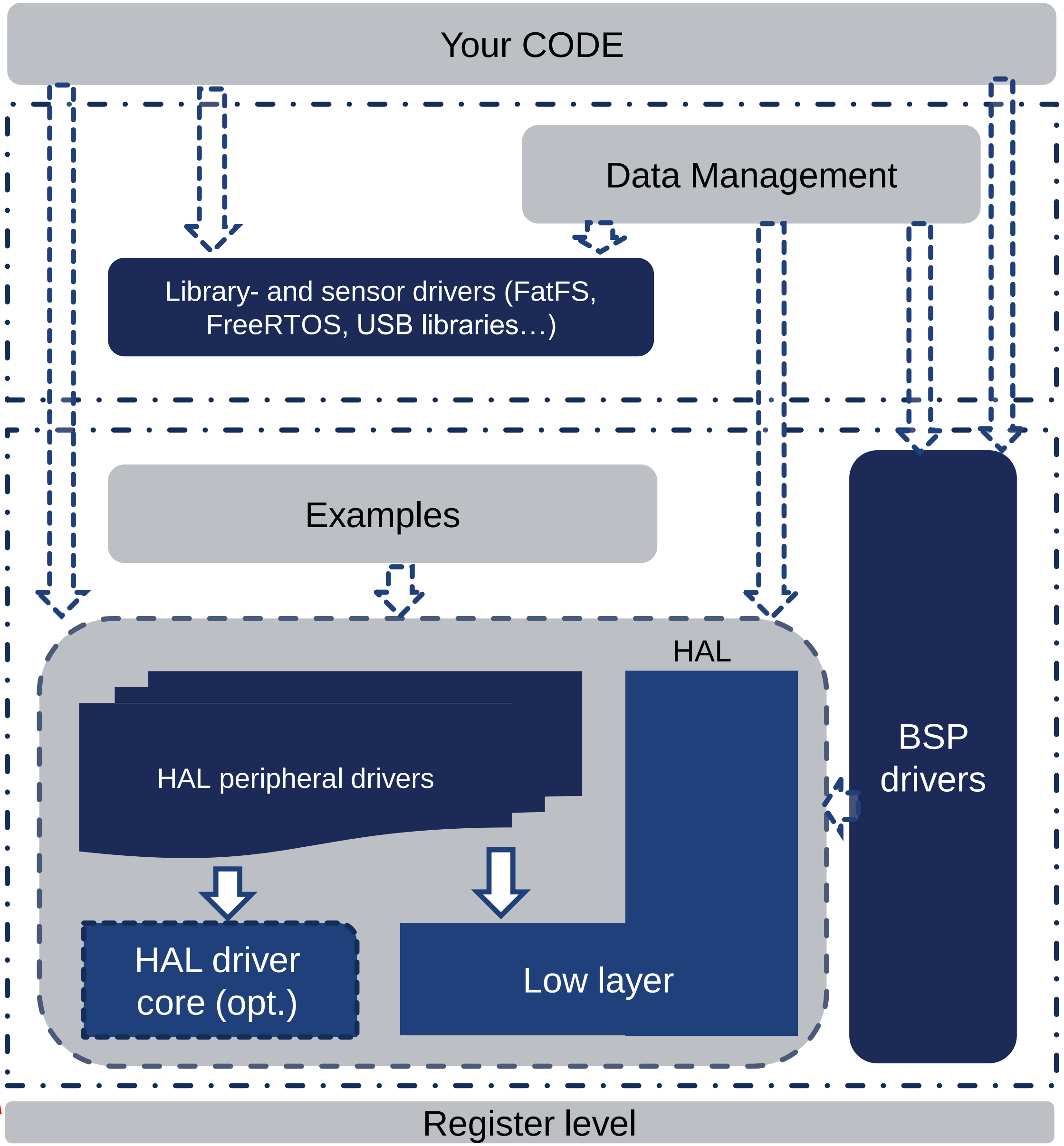
YOU

Application Level 2

Middleware Level 1

Drivers
HAL or LL Level 0

Advanced

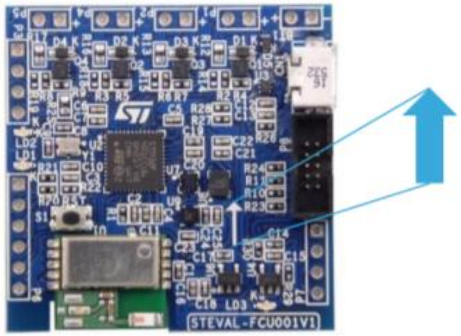


STWIN Template

Low-level drivers and data conversion is ready and working

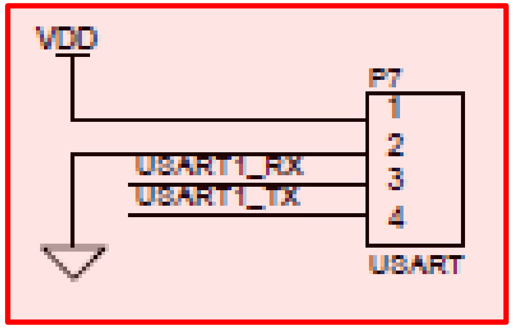
UART to USB

Serial connection



UART to USB

serial communication

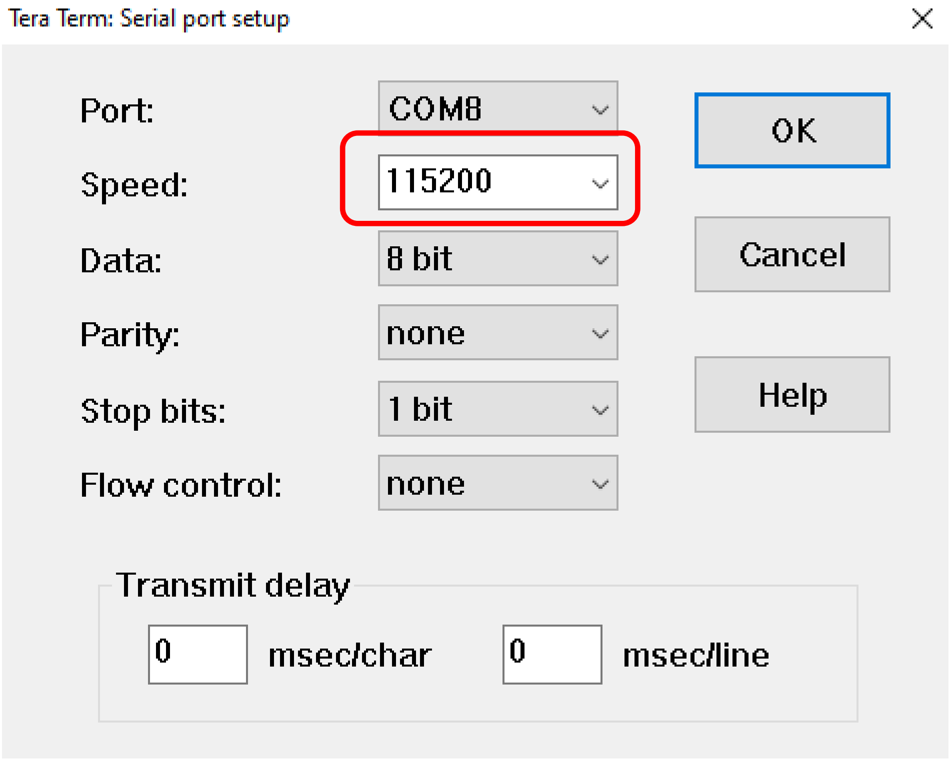


- USART1_RX to **TDX**
- USART1_TX to **RDX**
- GND - **GND**

Laptop configuration

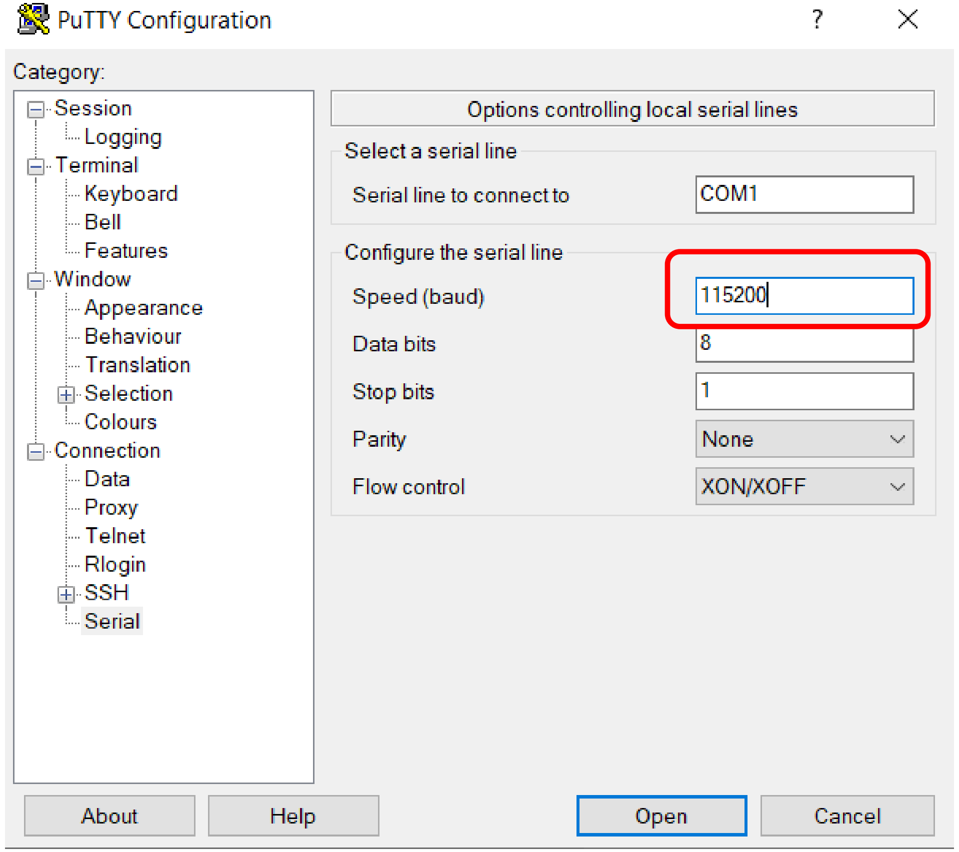
With Tera term:

Setup→ Serial port …



With Putty:

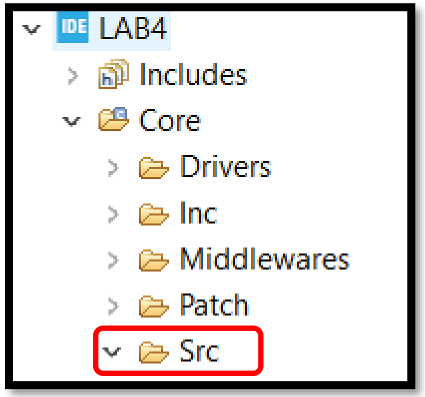
Connection→ Serial



Baudrate 115200 Hz

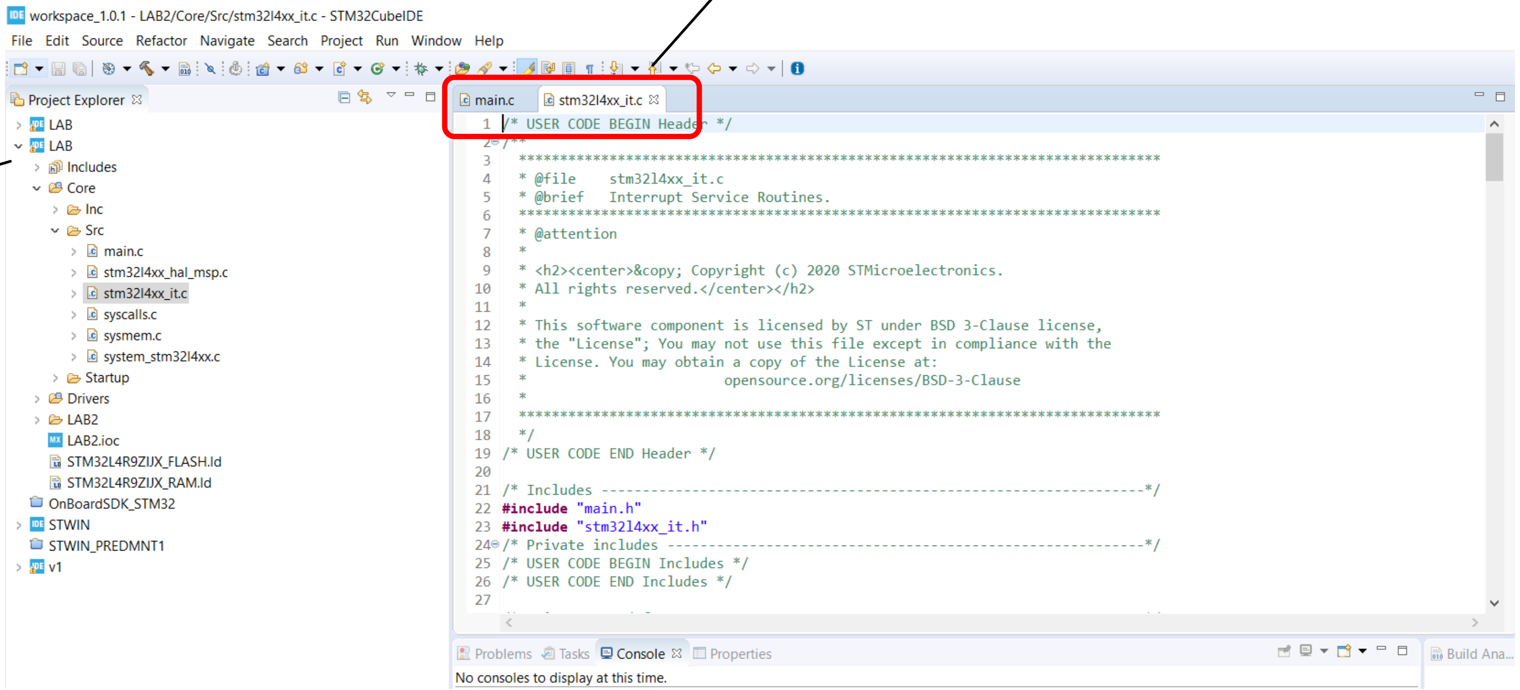
Preparation for LAB3

- Import the LAB3 template (name: LAB3)
- Open main.c
- Build it
- Read the application code….

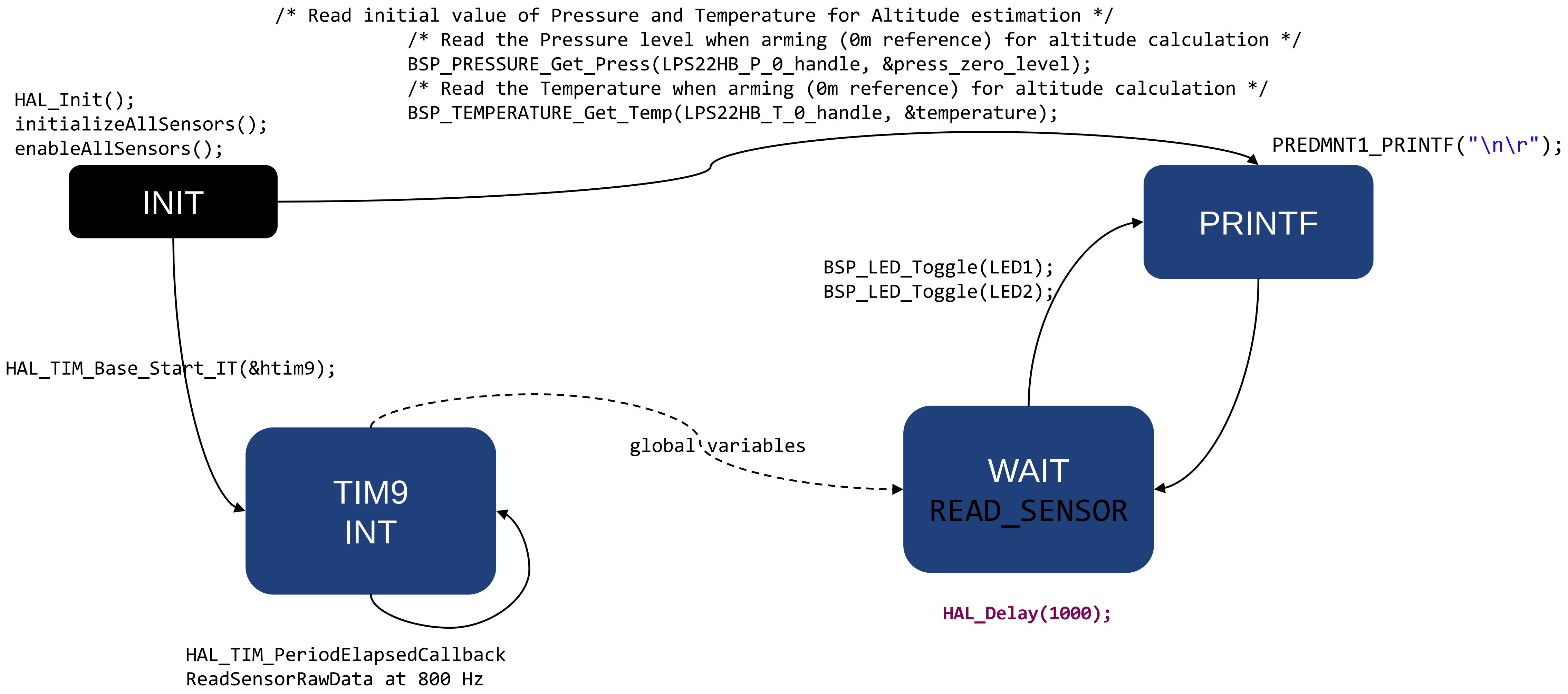


Project files

Project



LAB3 flow


/* Read initial value of Pressure and Temperature for Altitude estimation */
/* Read the Pressure level when arming (0m reference) for altitude calculation */
BSP_PRESSURE_Get_Press(LPS22HB_P_0_handle, &press_zero_level);
/* Read the Temperature when arming (0m reference) for altitude calculation */
BSP_TEMPERATURE_Get_Temp(LPS22HB_T_0_handle, &temperature);
HAL_Init();
initializeAllSensors();
enableAllSensors();
INIT
PREDMNT1_PRINTF("\n\r");
PRINTF
BSP_LED_Toggle(LED1);
BSP_LED_Toggle(LED2);
HAL_TIM_Base_Start_IT(&htim9);
global variables
TIM9
INT
WAIT
READ_SENSOR
HAL_Delay(1000);
HAL_TIM_PeriodElapsedCallback
ReadSensorRawData at 800 Hz

LAB3 Project application layer

```
/*
 * This function read sensor data and prepare data for proper coordinate system
 * according to definition of COORDINATE_SYSTEM
 * The unit of each data are:
 *      Acc - mg
 *      Gyro - mdps
 *      Mag - mguass
 */

void ReadSensorRawData(AxesRaw_TypeDef *acc, AxesRaw_TypeDef *gyro, AxesRaw_TypeDef *mag,
float *pre)
```

Read and print accelerometer data (x, y and z axes)

```
typedef struct {
    int32_t AXIS_X;
    int32_t AXIS_Y;
    int32_t AXIS_Z;
} AxesRaw_TypeDef;
```

- Global motion var. *acc, gyro, mag*
- Global pressure var. pressure **float**

Sensors value will be updated here at 800 Hz

LAB3 Template

```
/*
while (1)
{

    /* print the measured sensors at 1 Hz */
    /* sample rate is 800 Hz */
    PRINTF("************************************\n\r\n\r");
    PRINTF("ACC[x,y,z] %d mg %d mg %d mg \n\r", acc.AXIS_X,acc.AXIS_Y,acc.AXIS_Z);
    PRINTF("GYR[x,y,z] %d mdps %d mdps %d mdps \n\r", gyro.AXIS_X,gyro.AXIS_Y,gyro.AXIS_Z);
    PRINTF("MAG[x,y,z] %d mg %d mg %d mg \n\r", mag.AXIS_X,mag.AXIS_Y,mag.AXIS_Z);
    PRINTF("Pre: %f hPa \n\r", press);
    PRINTF("************************************\n\r\n\r");
```

YOUR CODE HERE

```
    //This function provides accurate delay (in milliseconds)
    HAL_Delay(1000);
    BSP_LED_Toggle(LED1);
    BSP_LED_Toggle(LED2);

}
/* USER CODE END 3 */
```

I2C2
- Accelerometer
- Gyroscope
- Magnetometer

- Hypsometric formula
- Pitch, Roll, Yaw

Read period

LAB3 Plot Template and code example

```c
/* altitude in meters */
float alt = getAltitude(press, temperature);
/* process the eCompass algorithm, calculate pitch, roll and yaw (azimuth) */
eCompass();
PRINTF("************************************\n\r\n\r");
PRINTF("Pitch: %f deg Roll: %f deg Yaw: %f deg \n\r", pitch,roll,azimuth);
PRINTF("Altitude: %f m \n\r", alt);
PRINTF("************************************\n\r\n\r");
```

- Calculate altitude from hypsometric formula

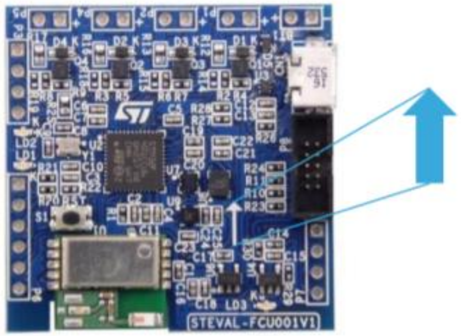
Plot format, all variables are floating point numbers

https://www.codingunit.com/printf-format-specifiers-format-conversions-and-formatted-output

Serial connection

UART

serial communication



USB
UNIVERSAL SERIAL BUS

```
****************************************

ACC[x,y,z] 44 mg 23 mg 1008 mg
GYR[x,y,z] 700 mdps -70 mdps -70 mdps
MAG[x,y,z] 11 mg 200 mg -377 mg
Pre: 977.400024 hPa
****************************************
```

```
STEVAL- STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

²STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

STEVAL-FCU001V1 FW rev.1.0 - Sep 2017

LSM6DSL MEMS Accelerometer initialized and enabled
LSM6DSL MEMS Gyroscope initialized and enabled
LIS2MDL Magnetometer initialized and enabled
LPS22HB Pressure sensor initialized and enabled
LPS22HB Temperature sensor initialized and enabled
LAB3 Init...
```

LAB3: Exercise overview

Template can be found in your Polybox folder LAB3

Exercise	Assignment	Concept
Exercise 1	Import and execute the template. You should read the sensor values on the terminal	SPI, I2C, UART, USB
Exercise 2	Estimate the altitude from the pressure sensor. Plot it in the proposed format. Set the reference (0 m) with respect the room floor	SPI, I2C, UART, USB
Exercise 3	Calculate Pitch, Roll and Yaw (see next slides). Plot it in the proposed format.	
Exercise 3	Enable LED1 when the board is > 80° on the x axis.	SPI, I2C, UART, USB
Exercise 4	Enable LED2 when the board is > 80° on y axis	GPIO, TIMER, LPM
Exercise 5 (advanced)	Magnetometer hard-iron calibration and accuracy comparison with a reference platform. Plot the calibrated Yaw value.	GPIO, TIMER, LPM

LAB3: Exercise overview

Template can be found in your Polybox folder LAB3

Exercise	Assignment	Concept
Question 1	At which speed is configured the SPI?	SPI, I2C, UART, USB
Question 2	Why we use a baudrate of 115200 on Tera term (Putty) ?	SPI, I2C, UART, USB
Question 3	Pitch and Roll instability (division by 0). When it is possibile? How we avoided this issue? (from theoretical readings)	

eCompass math

$$pitch = atan2\left(\frac{-G_X}{\sqrt{(G_Y^2 + G_Z^2)}}\right)$$

$$roll = atan2\left(\frac{G_Y}{\sqrt{(\mu G_X^2 + G_Z^2)}}\right)$$

Equation for the roll angle is an approximation but has several characteristics that make it attractive:

- It is impossible for both numerator and denominator to be simultaneously zero and give an undefined or unstable estimate of the roll angle.
- smoothly drives the roll angle to zero as the drone becomes oriented vertically upwards or downwards since Gx approaches 1g and Gy and Gz approach zero
- µ between 0.01 and 1, standard value is 1
- Accelerometer data cannot be used for tilt measurement if high-g is ongoing. Accelerometer data can only be used when the modulus is near *g*: $G_x^2 + G_Y^2 + G_Z^2 = 1$, when Gx / Gy / Gz are expressed in *g*.

- **Gx = accelerometer value on x axis**

eCompass math

$$By2 = B_Z \sin(Roll) - B_Y \cos(Roll)$$

$$Bz2 = B_Y \sin(Roll) + B_Z \cos(Roll)$$

$$Bx3 = B_X \cos(Pitch) + Bz2 \sin(Pitch)$$

$$Yaw = atan2\left(\frac{By2}{Bx3}\right)$$

compute eCompass from tilt-compensated (yaw angle) magnetometer data:

- Magnetometer data cannot be used for the eCompass (Psi) if hard/soft iron effects are not compensated and/or if magnetic anomalies are present. **Hard-iron** effects are offsets that must be subtracted from Bx / By / Bz.

- **Bx = magnetometer value on x axis**

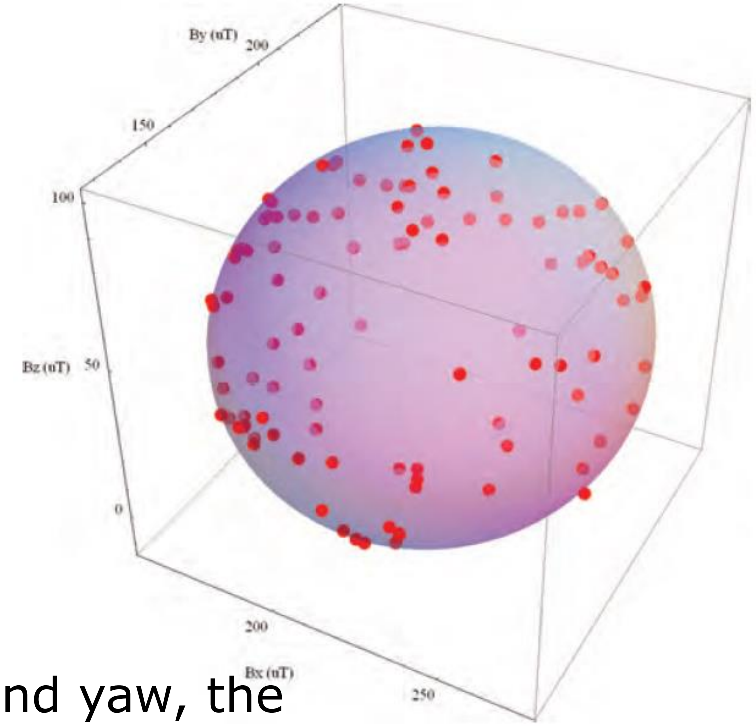
- Rad to degree conversion (for plotting on the terminal)

$$Yaw = Yaw * \left(\frac{180}{\pi}\right)$$

$$Pitch = Pitch * \left(\frac{180}{\pi}\right)$$

$$Roll = Roll * \left(\frac{180}{\pi}\right)$$

Magnetometer hard-iron compensation



$$(B_x - V_x)^2 + \left(B_y - V_y\right)^2 + (B_z - V_z)^2 = B_0^2$$

Equation simply states that under arbitrary rotations in roll, pitch, and yaw, the magnetometer readings lie on the surface of a sphere with radius B_0 centered at the **hard-iron interference** V_X, V_Y, and V_Z. **(it is a static offset! –SIMPLE!)**

To obtain your own hard-iron calibration, simply record and plot the magnetometer readings under random orientations and estimate your own hard-iron correction V_X, V_Y, V_Z from the center of the resulting sphere. A simple but effective technique is to rotate the eCompass in a figure of eight twisting motions for a few seconds, record the minimum and maximum magnetometer readings, and compute the hard-iron calibration from their average

hard-iron compensation: C implementation

$$\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = \frac{1}{2}\begin{pmatrix} \max(B_x) + \min(B_x) \\ \max(B_y) + \min(B_y) \\ \max(B_z) + \min(B_z) \end{pmatrix}$$

```c
AxesRaw_TypeDef mag_min = {10000,10000,10000}, mag_max = {-10000,-10000,-10000};
void MagCalibration ( void ){

    /* find min */
    if (mag.AXIS_X < mag_min.AXIS_X) mag_min.AXIS_X = mag.AXIS_X;
    if (mag.AXIS_Y < mag_min.AXIS_Y) mag_min.AXIS_Y = mag.AXIS_Y;
    if (mag.AXIS_Z < mag_min.AXIS_Z) mag_min.AXIS_Z = mag.AXIS_Z;
    /* find max */
    if (mag.AXIS_X > mag_max.AXIS_X) mag_max.AXIS_X = mag.AXIS_X;
    if (mag.AXIS_Y > mag_max.AXIS_Y) mag_max.AXIS_Y = mag.AXIS_Y;
    if (mag.AXIS_Z > mag_max.AXIS_Z) mag_max.AXIS_Z = mag.AXIS_Z;
    /* hard-iron calibration */
    mag.AXIS_X -= (mag_max.AXIS_X + mag_min.AXIS_X)/2;
    mag.AXIS_Y -= (mag_max.AXIS_Y + mag_min.AXIS_Y)/2;
    mag.AXIS_Z -= (mag_max.AXIS_Z + mag_min.AXIS_Z)/2;

}
```

MagCalibration must be executed for every sample (at 800 Hz) after the **ReadSensorRawData** function in **HAL_TIM_PeriodElapsedCallback**

math.h

- For math function in C follow the guide:
  https://www.tutorialspoint.com/c_standard_library/math_h.htm
- You will need: Sin, Cos, atan2, Pow