STM32 IoT Arduino

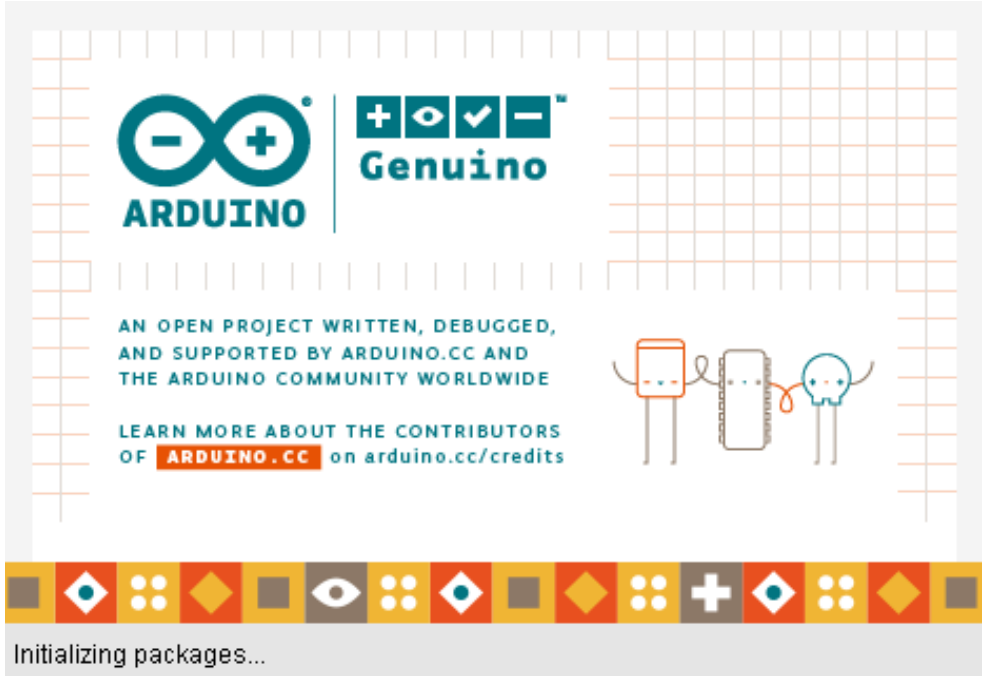
Festival Transfo 2018

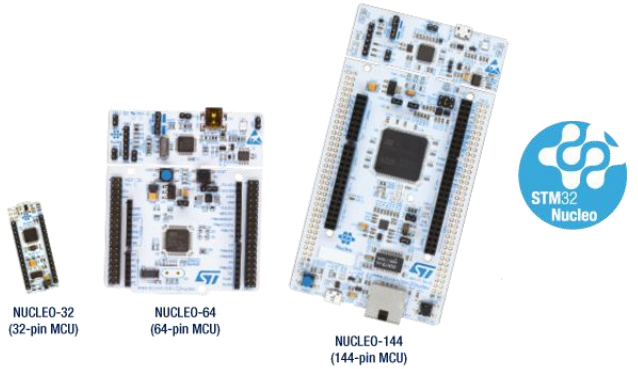
life.augmented

transfo
le numérique et nous

Arduino VS STM32

- Most of STM32 boards are compliant with Arduino Eco-system.
- Many libraries/examples are available.





Discovery

life.augmented

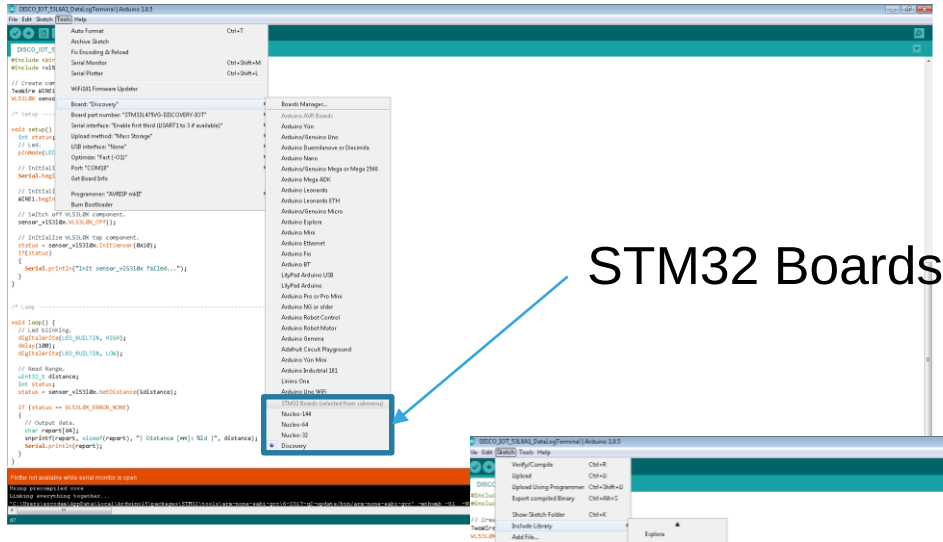
Internet of Things

STM32 ODE on stm32duino

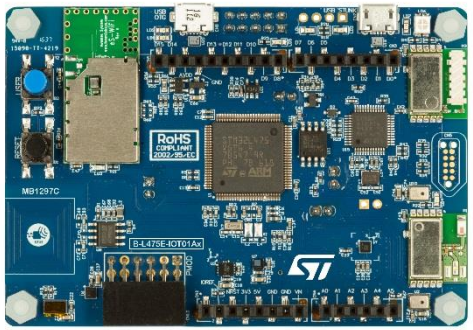
- Nucleo and Discovery boards supported
  - NUCLEO-L476RG, -F401RE, -F411RE, -L053R8, -L152RE, -F030R8, -F091RC, -F103RB, -F207ZG, -F303RE, -F429ZI, -L432KC
  - STM32VLDISCOVERY, STM32F407G-DISC1, STM32F746G-DISCOVERY, B-L072Z-LRWAN1 and **B-L475E-IOT01A**
- X-Nucleo boards supported
  - X-NUCLEO-IKS01A1 (+ LSM6DS3), X-NUCLEO-IKS01A2
  - X-NUCLEO-NFC01A1, X-NUCLEO-NFC04A1
  - X-NUCLEO-6180XA1, X-NUCLEO-53L0A1
  - X-NUCLEO-IDB05A1 (only Device Mode)
  - X-NUCLEO-LED61A1
  - X-NUCLEO-IKA01A1
  - X-NUCLEO-IHM02A1

Where you can find these libraries
http://stm32duino.com/viewtopic.php?f=60&t=2902
https://github.com/stm32duino/Arduino_Core_STM32

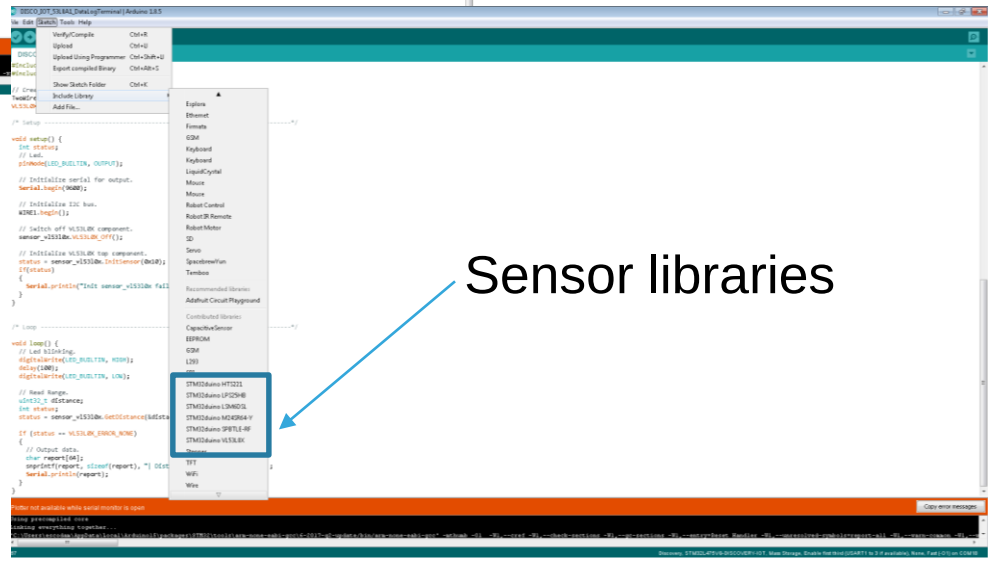
Arduino for STM32



STM32 Boards



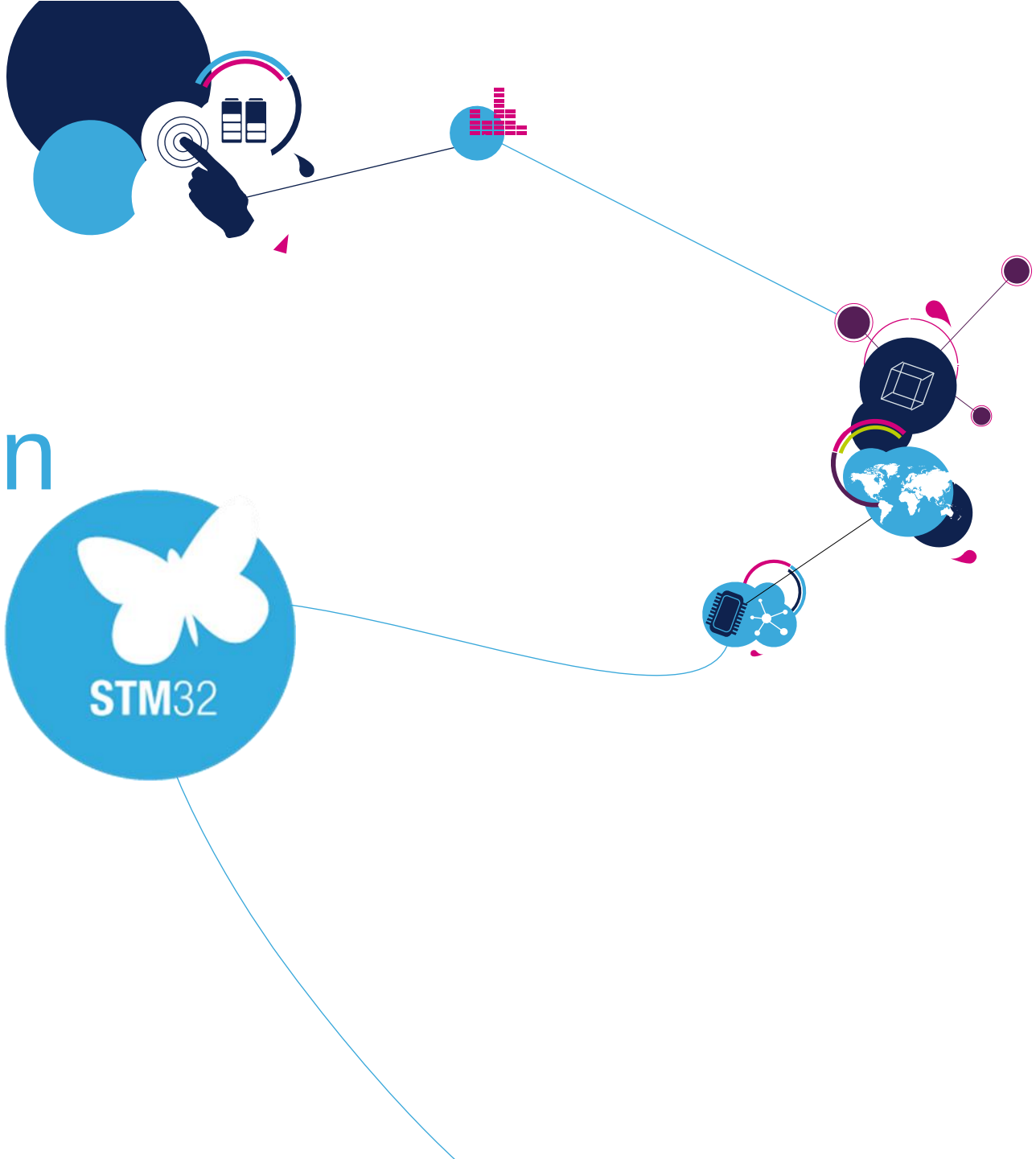
B-L475E-IOT01AX board ready for IoT



Sensor libraries

STM32 DUINO

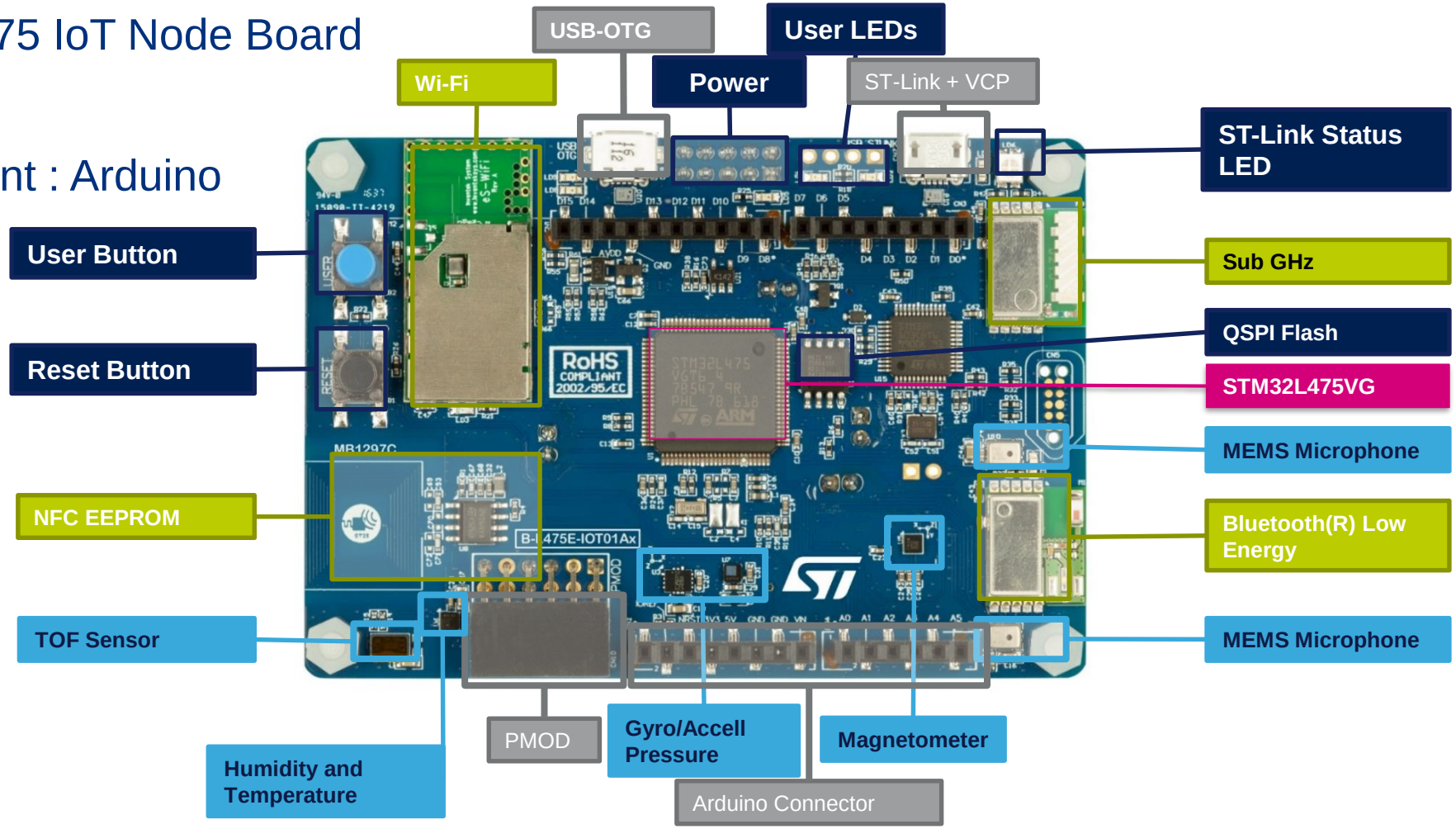
Board presentation



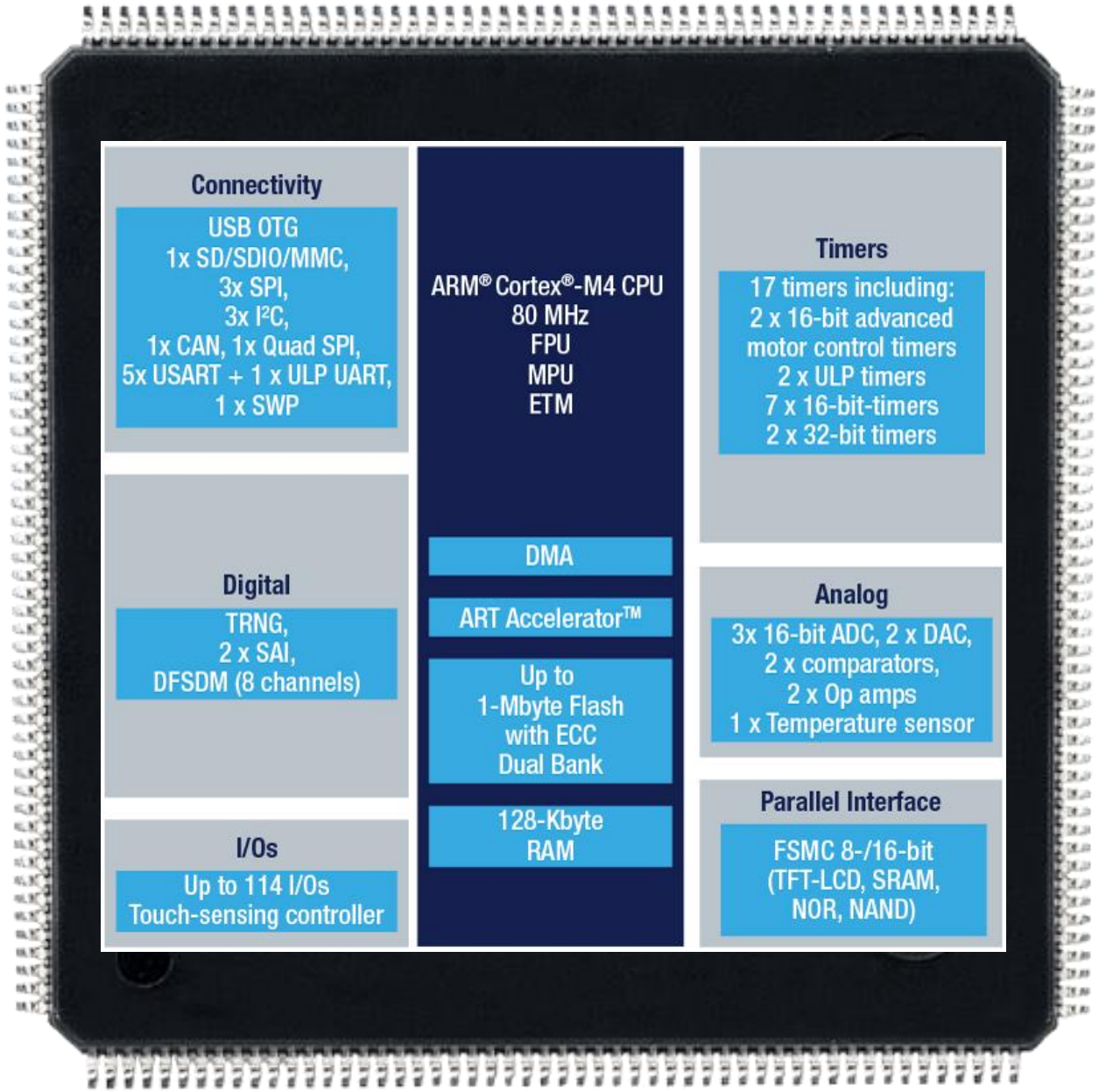
life.augmented

Festival Transfo

- Board presentation
  - Board : L475 IoT Node Board
  - Environment : Arduino



STM32L475



- STM32L4x devices are the ultra-low-power microcontrollers
  - 100 µA/MHz run mode
  - 1.1 µA Stop 2 mode,
  - 1.4 µA with RTC
  - 120 nA Standby mode
  - 30 nA Shutdown mode

life.augmented

Festival Transfo – Pre requis

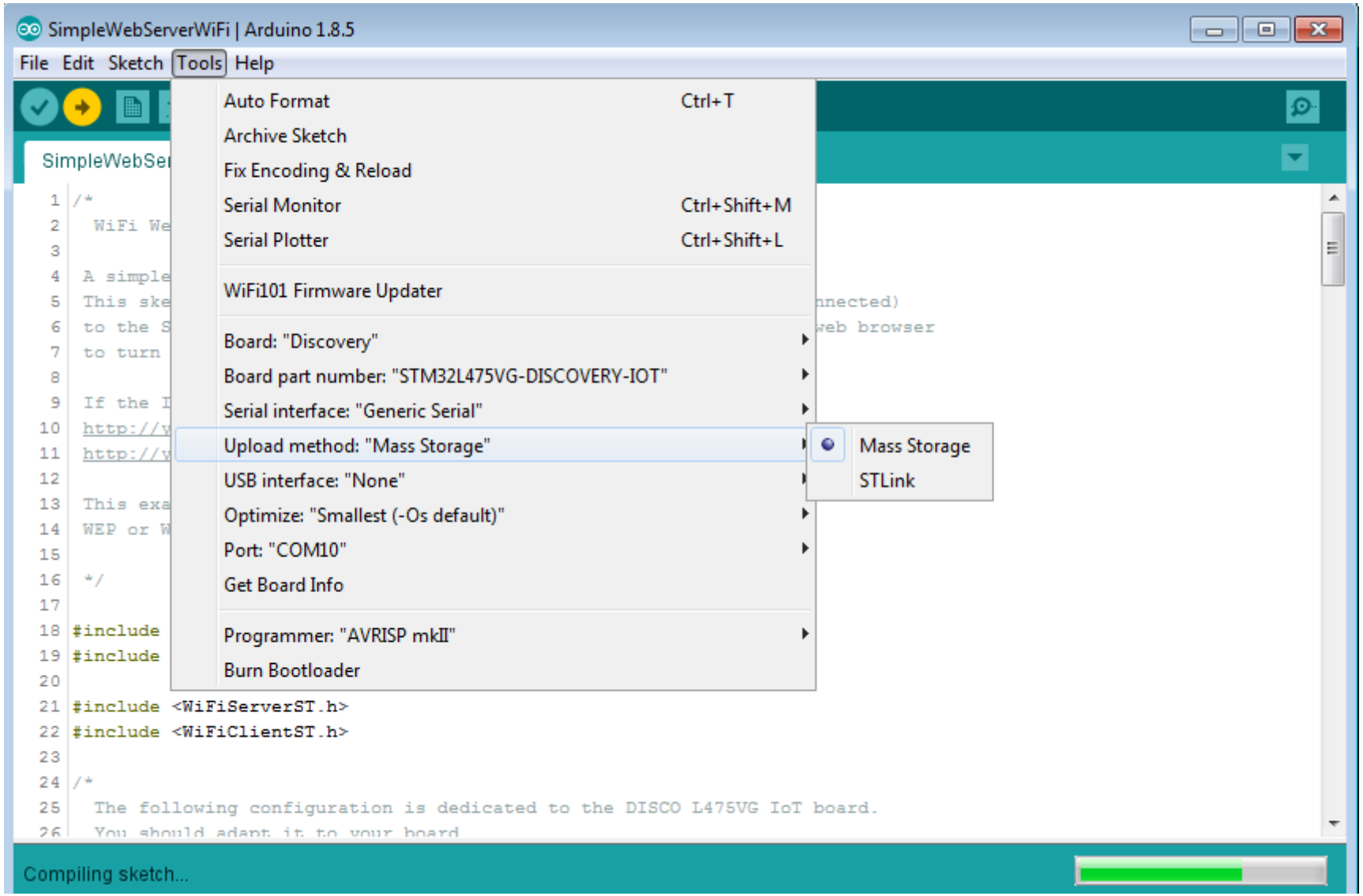
Pour utiliser les cartes STM32 (Nucleo32, Nucleo64, Nucleo144 et Discovery) dans l'environnement Arduino vous devez :

- Connection internet
- Board IoT node connecté au PC via USB
- Sur PC, installation de la derniere version de l'IDE Arduino disponible ici : https://www.arduino.cc/en/main/software (Linux/Windows)
    - Ajouter des cartes STM32 Arduino dans le « Board Manager » : https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/master/STM32/package_stm_index.json
    - Ajout des libraries Arduino pour la périphériques de la carte IoT
- Sur Smartphone/Tablette, installation des applications : BLUE NRG, NFC Tools

Festival Transfo – Arduino IDE

- Installation de la derniere version de l'IDE Arduino disponible ici : https://www.arduino.cc/en/main/software (Linux/Windows)
- Rajouter le support des cartes Arduino STM32
  - https://raw.githubusercontent.com/stm32duino/BoardManagerFiles/master/STM32/package_stm_index.json
- Selectionner la Nucleo L475IoT dans le « Board Manager »
- Selectionner l'interface UART dans Tools->Port <Image>
  - Sur Mac, on trouve /dev/tty.usbmodem-1511
  - Sur Windows, un nouveau COM port.. Par example, COM5
  - Sur Linux, une entrée du style /dev/ttyACM0
- Methode de programmation (STLINK ou Mass storage) <Image>

Arduino IDE presentation



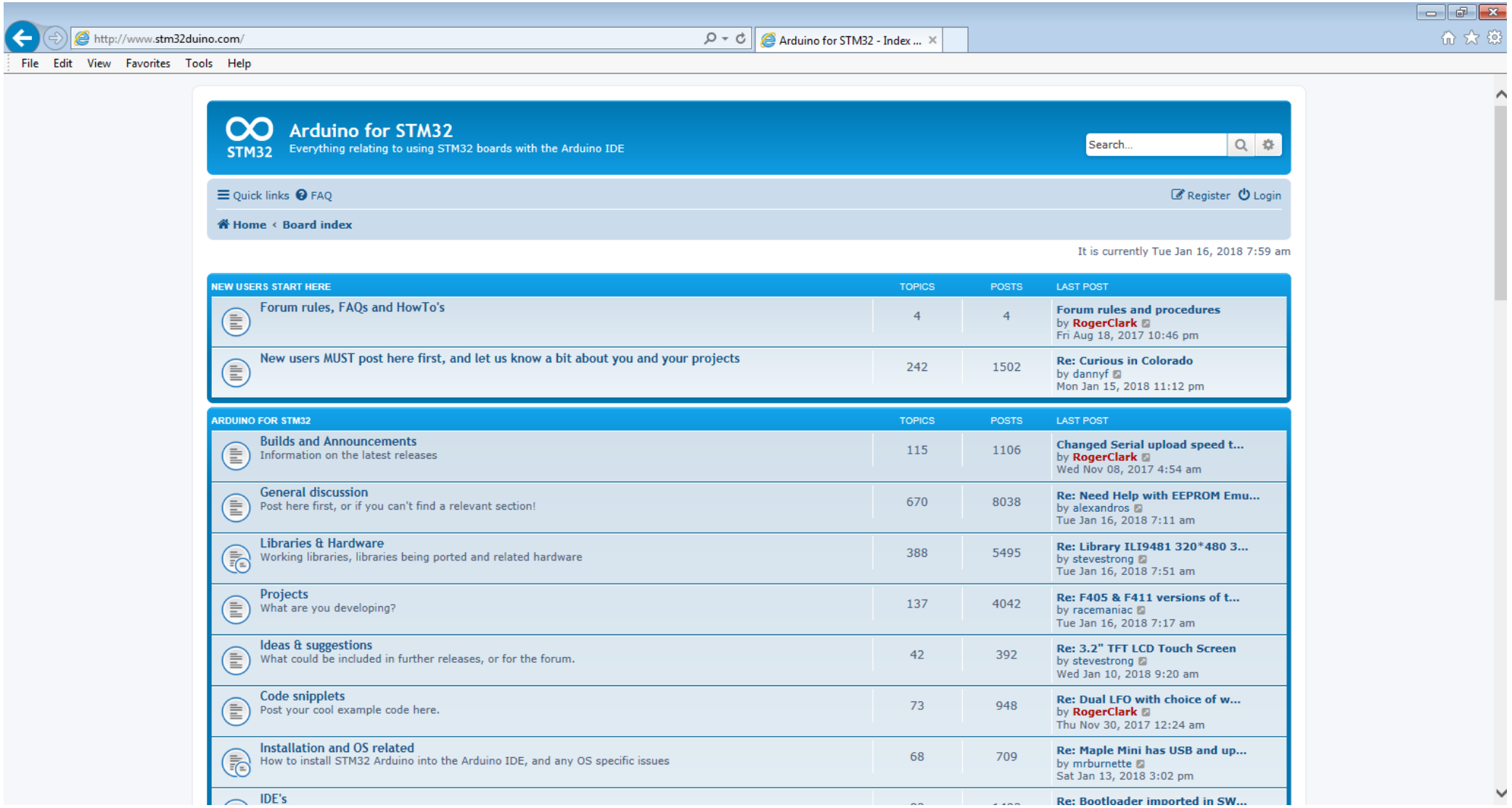
life.augmented

Festival Transfo – Arduino libraries

- Dans Sketch -> Include libraries -> Manage Libraries
- Rechercher avec le terme “STM32duino” et ajouter les librairies suivantes:
  - HTS221 (Humidity/Temperature)
  - ISM43362-M3G-L44 (Wifi)
  - LIS3MDL (3D magnetometre)
  - LPS25HB (Barometer)
  - LSM6DSL (Accelometer/Gyroscope)
  - M24SR64-Y (NFC/RFID)
  - SPBTLE-RF (BlueNRG)
  - VL53L0X (Time-Of-Flight)

Checkpoint #1
Libraries installation

STM32duino Forum



Festival Transfo – Applications

- BlueNRG :

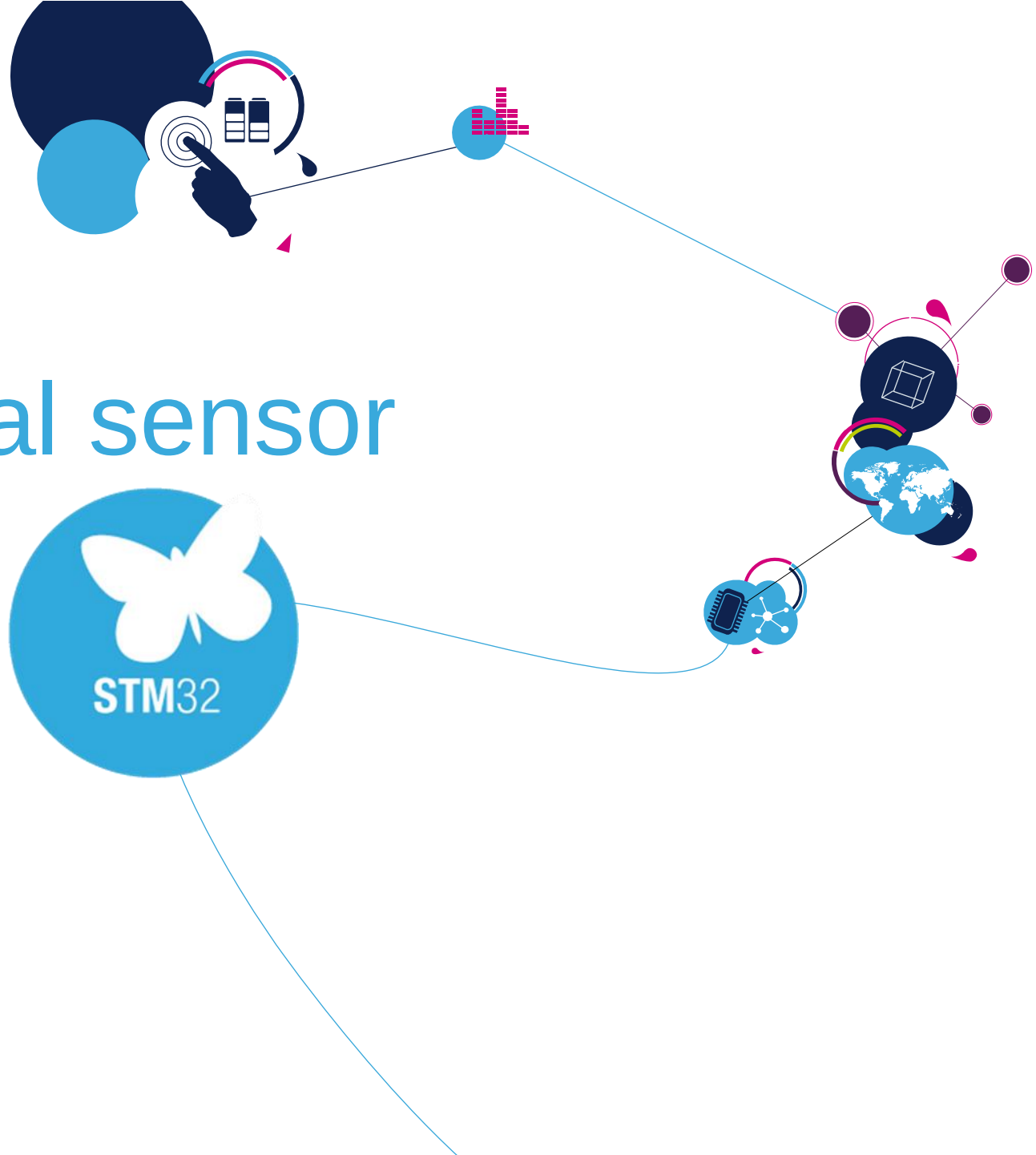
  App allows you to access all the sensor data directly from your mobile device via the Bluetooth® Low Energy protocol

- NFC Tools :

  App allows you to read and write NFC tags

Hands On Thermal sensor

HTS221 (Humidity/Temperature)



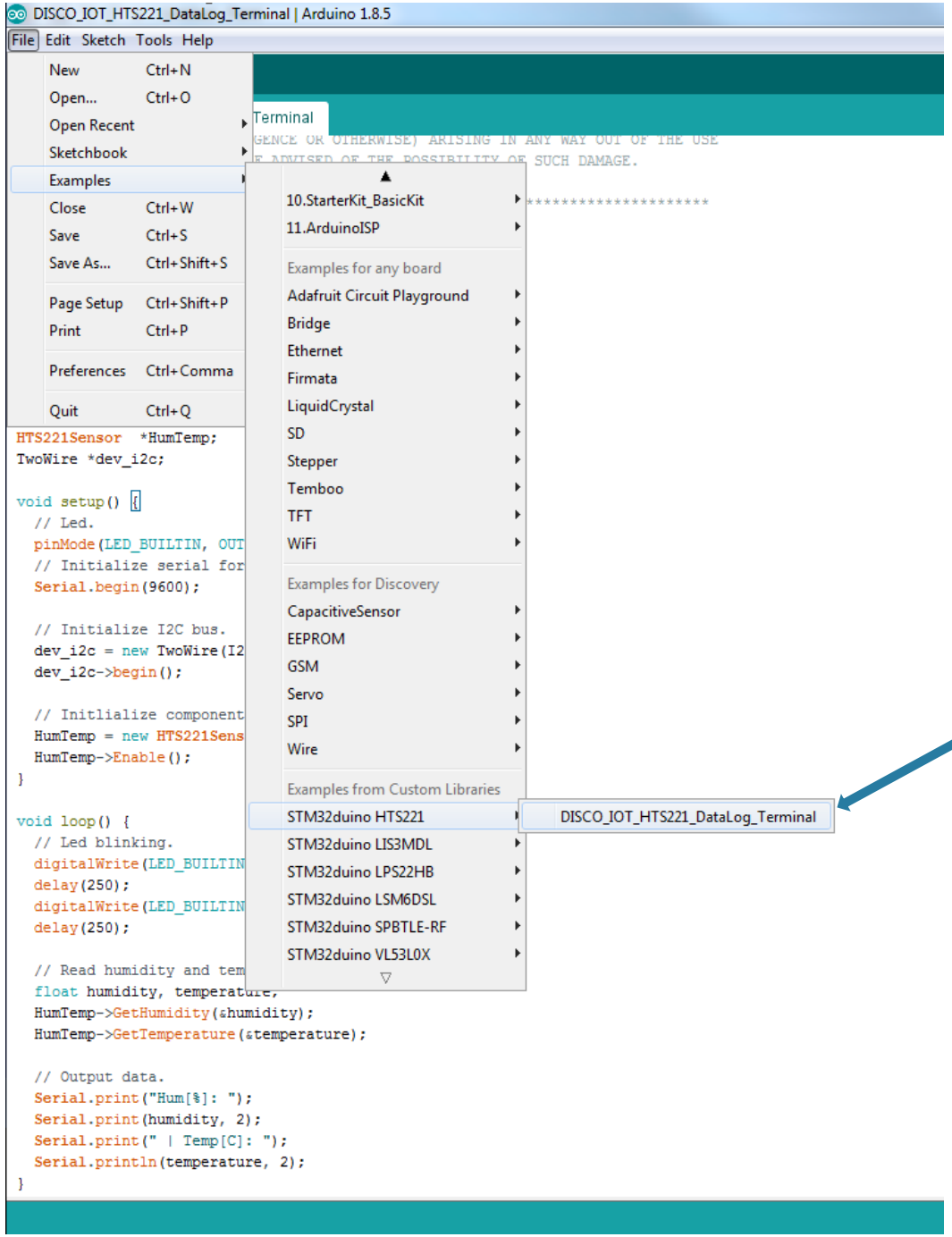
life.augmented

Thermal Sensor

- Objectif:
  - Lire les valeurs d'humidité et de température du HTS221 se trouvant sur la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer ces valeurs et les afficher sur le terminal
- Outils:
  - ARDUINO IDE et serial monitor
  - Librairie « STM32duino_HTS221 »
  - Exemple " DISCO_IOT_HTS221_DataLog_Terminal "

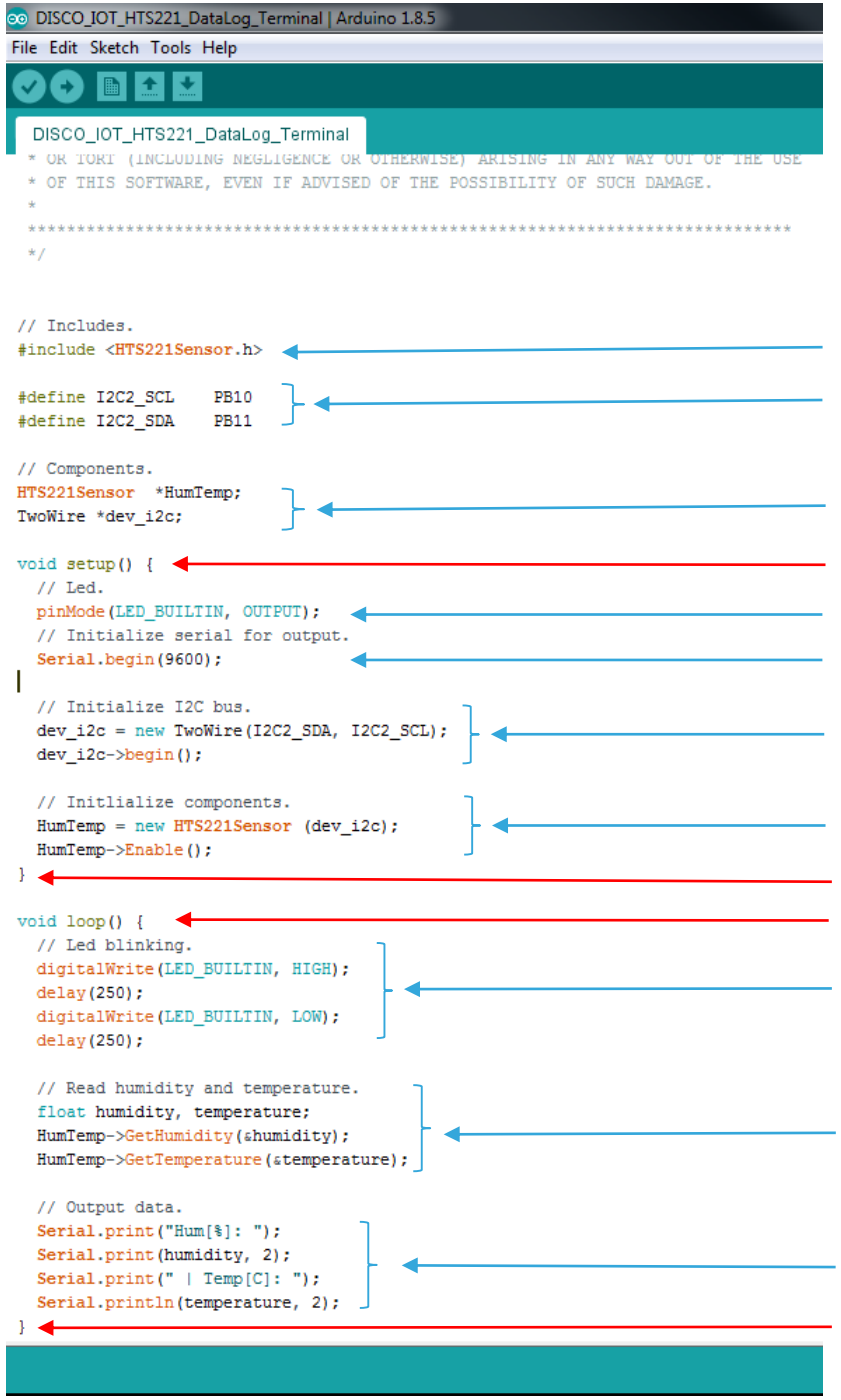
Thermal Sensor

- Télécharger la librairie « STM32duino_HTS221 »
- Ouvrir l'exemple « DISCO_IOT_HTS221_DataLog_Terminal »



life.augmented

Thermal Sensor



Librairie des fonctions du HTS221

Définition des broches du bus I2C

Instantiation des classes HTS221 et I2C

Début de la fonction d'initialisation

Initialisation du port en sortie pour la LED

Initialisation de l'UART a 9600 BAUD

Initialisation du bus I2C

Initialisation du HTS221

Fin de la fonction d'initialisation

Début de la fonction de la boucle principale

Clignotement de la LED

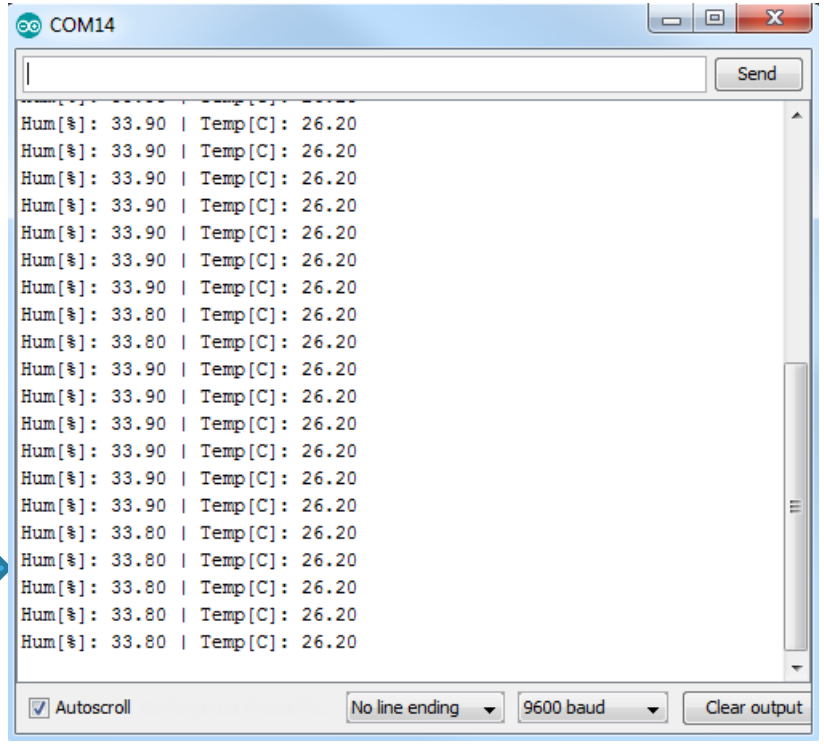
Lecture de l'humidité et de la température provenant du HTS221

Ecriture de l'humidité et de la température sur le terminal via l'UART

Fin de la fonction de la boucle principale

Thermal Sensor

- Ouvrir “serial monitor” pour espionner la lecture/ecriture

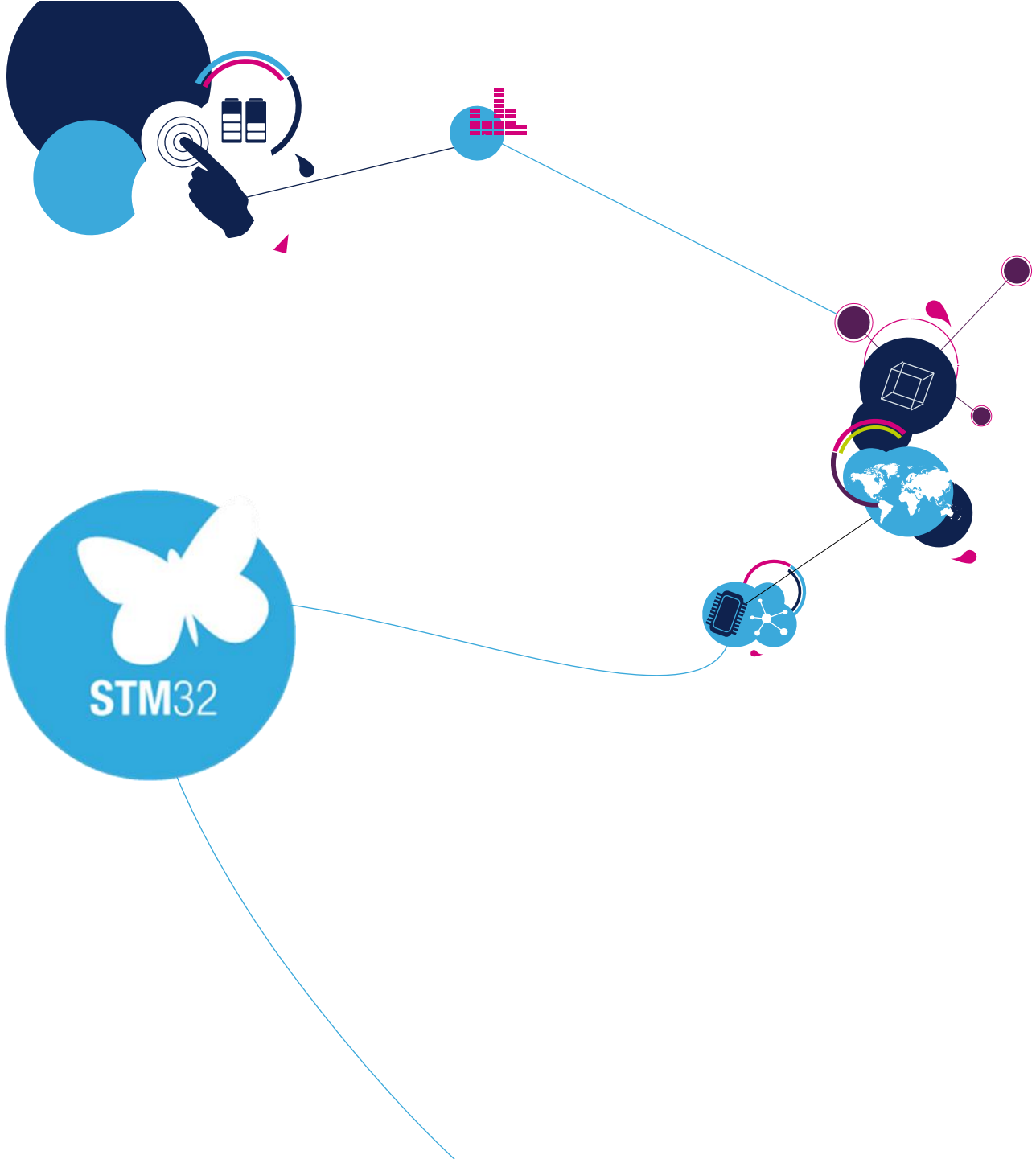


Checkpoint # 2
Thermal Sensor data

life.augmented

Hands On MEMs

LIS3MDL (3D magnetometer)



life.augmented

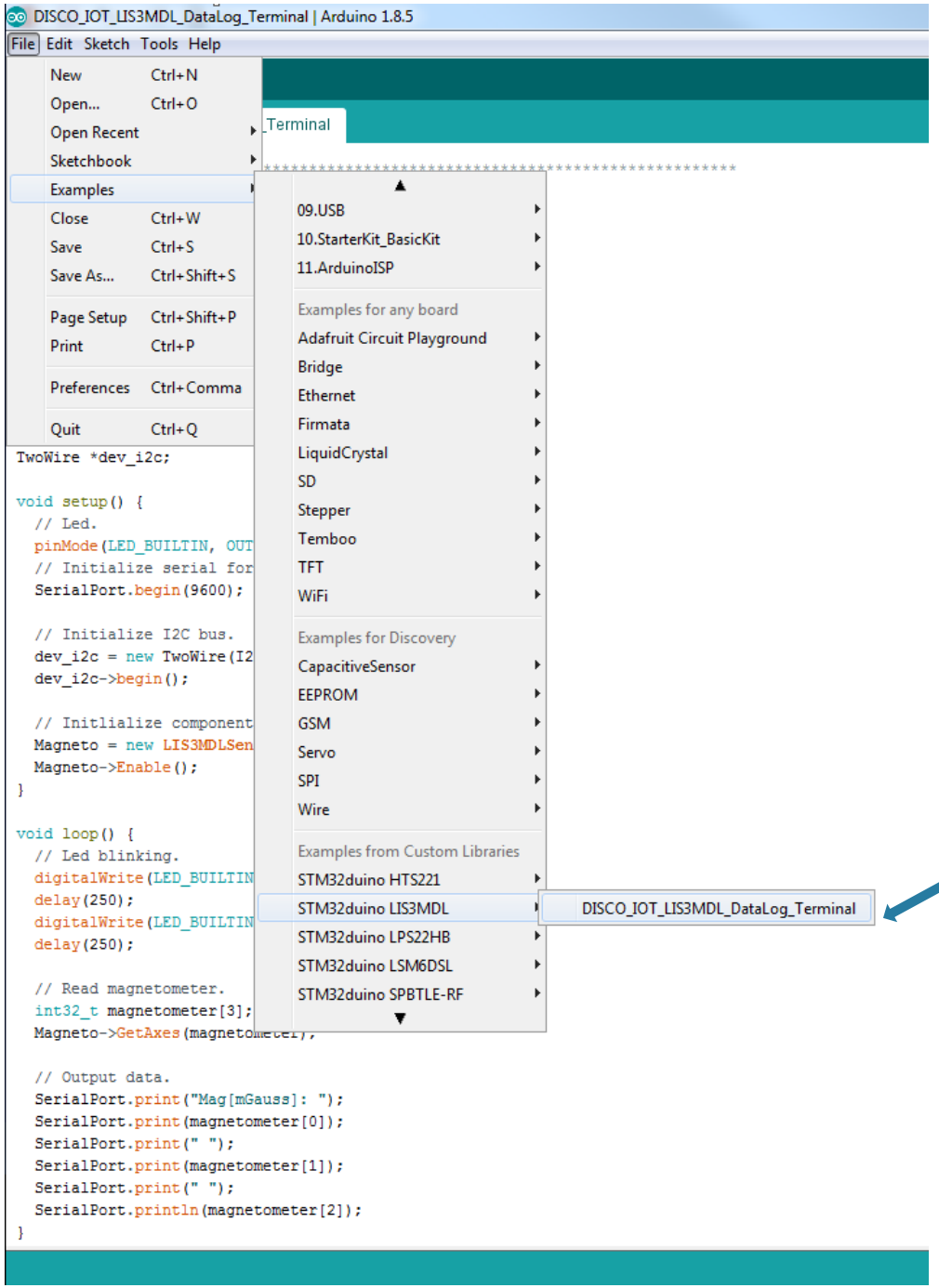
3D magnetometer

- Objectif:
  - Lire les valeurs des 3 axes du magnétomètre (intensité du champs magnétique) du LIS3MDL se trouvant sur la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer ces valeurs et les afficher sur le terminal

- Outils:
  - ARDUINO IDE et serial monitor
  - Librairie « STM32duino_LIS3MDL »
  - Exemple "DISCO_IOT_LIS3MDL_DataLog_Terminal "

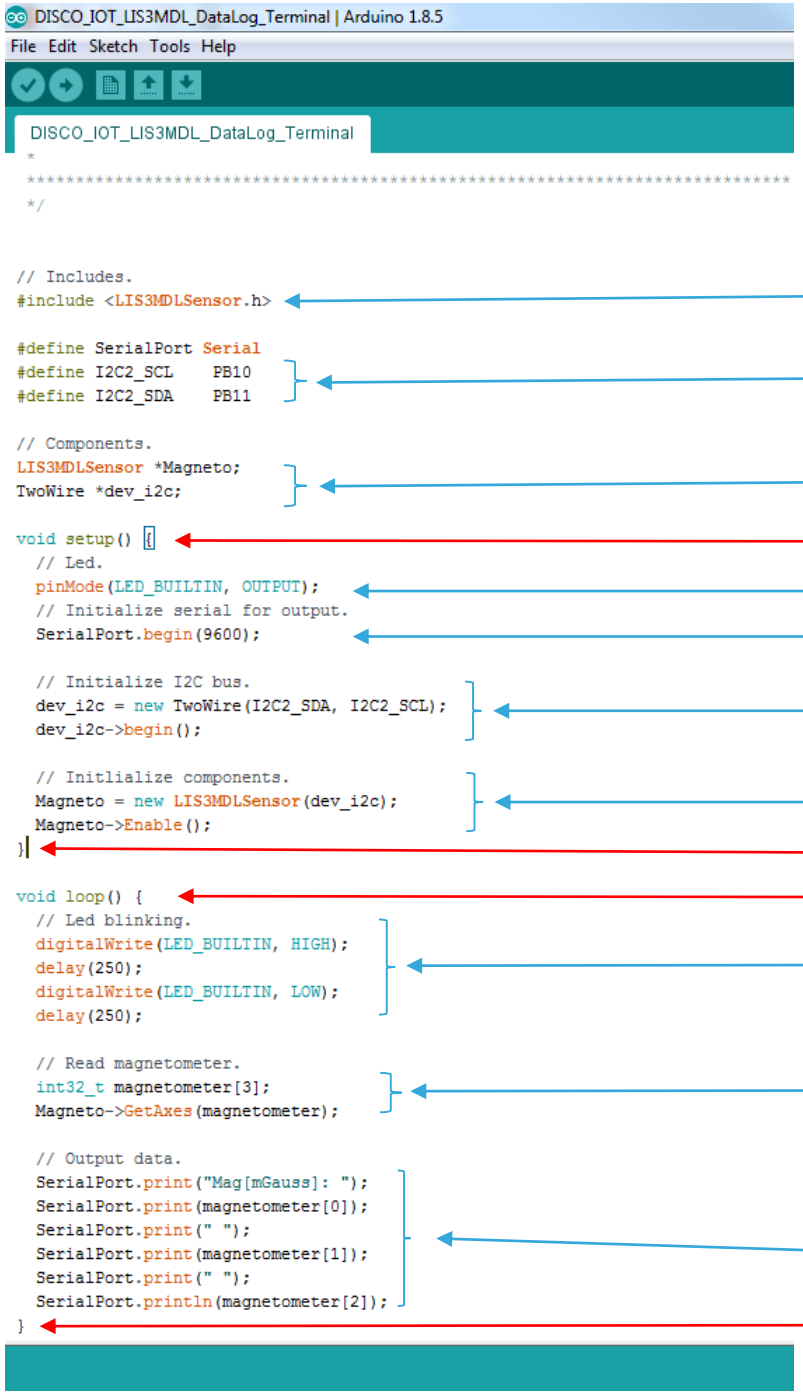
3D magnetometer

- Télécharger la librairie « STM32duino_HTS221 »
- Ouvrir l'exemple « DISCO_IOT_LIS3MDL_DataLog_Terminal »



life.augmented

3D magnetometer



Librairie des fonctions du LIS3MDL

Définition des broches du bus I2C

Instantiation des classes LIS3MDL et I2C

Début de la fonction d'initialisation

Initialisation du port en sortie pour la LED

Initialisation de l'UART a 9600 BAUD

Initialisation du bus I2C

Initialisation du LIS3MDL

Fin de la fonction d'initialisation

Début de la fonction de la boucle principale

Clignotement de la LED

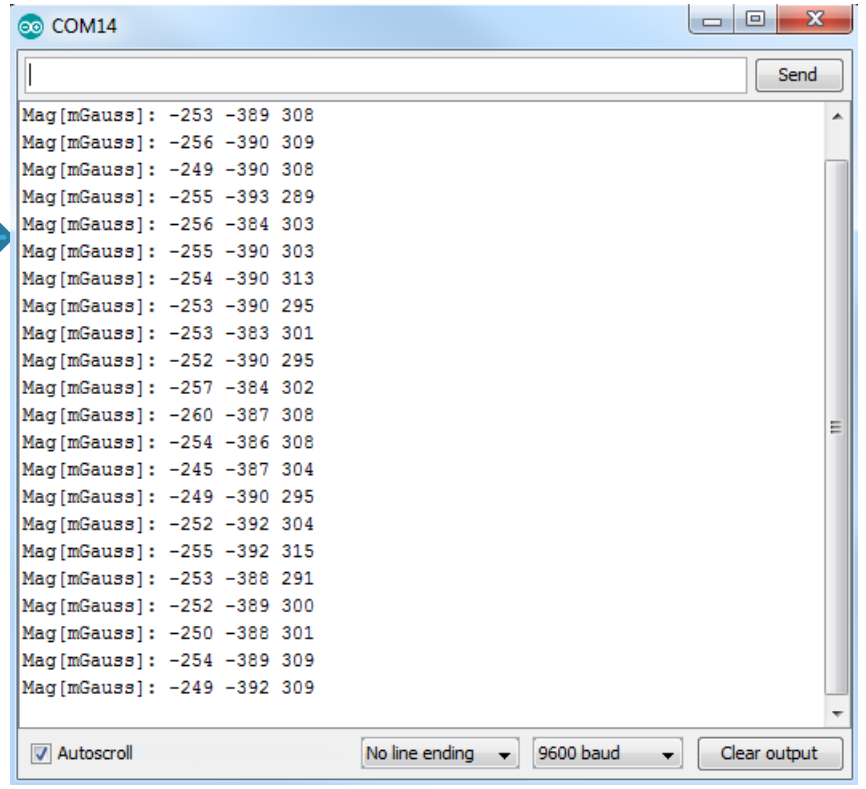
Lecture des 3 axes du LIS3MDL

Ecriture des valeurs des 3 axes sur le terminal via l'UART

Fin de la fonction de la boucle principale

3D magnetometer

- Ouvrir “serial monitor” pour espionner la lecture/ecriture

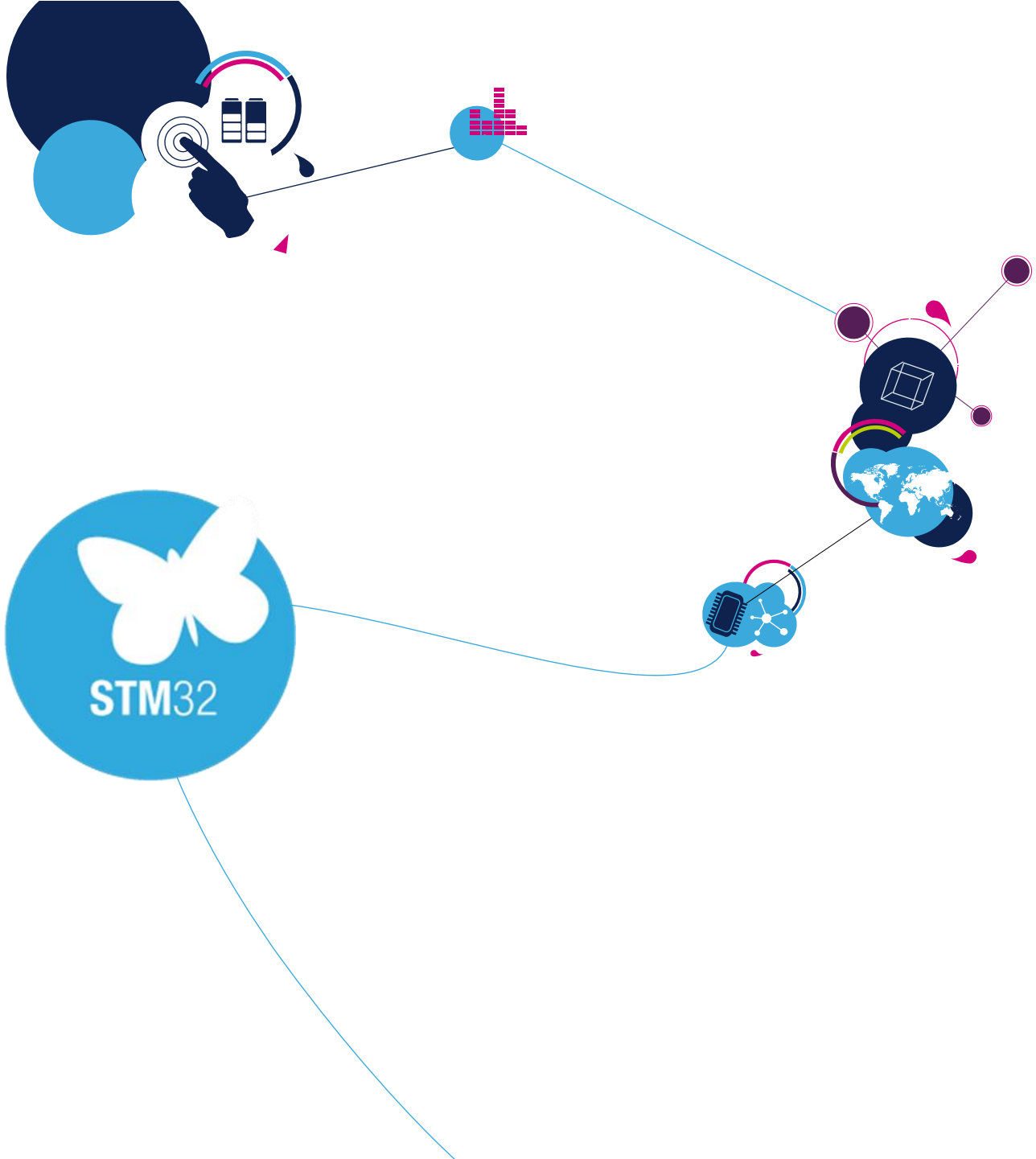


Checkpoint # 3
Magnetometer data

life.augmented

Hands On MEMs

LPS25HB (Barometer)



life.augmented

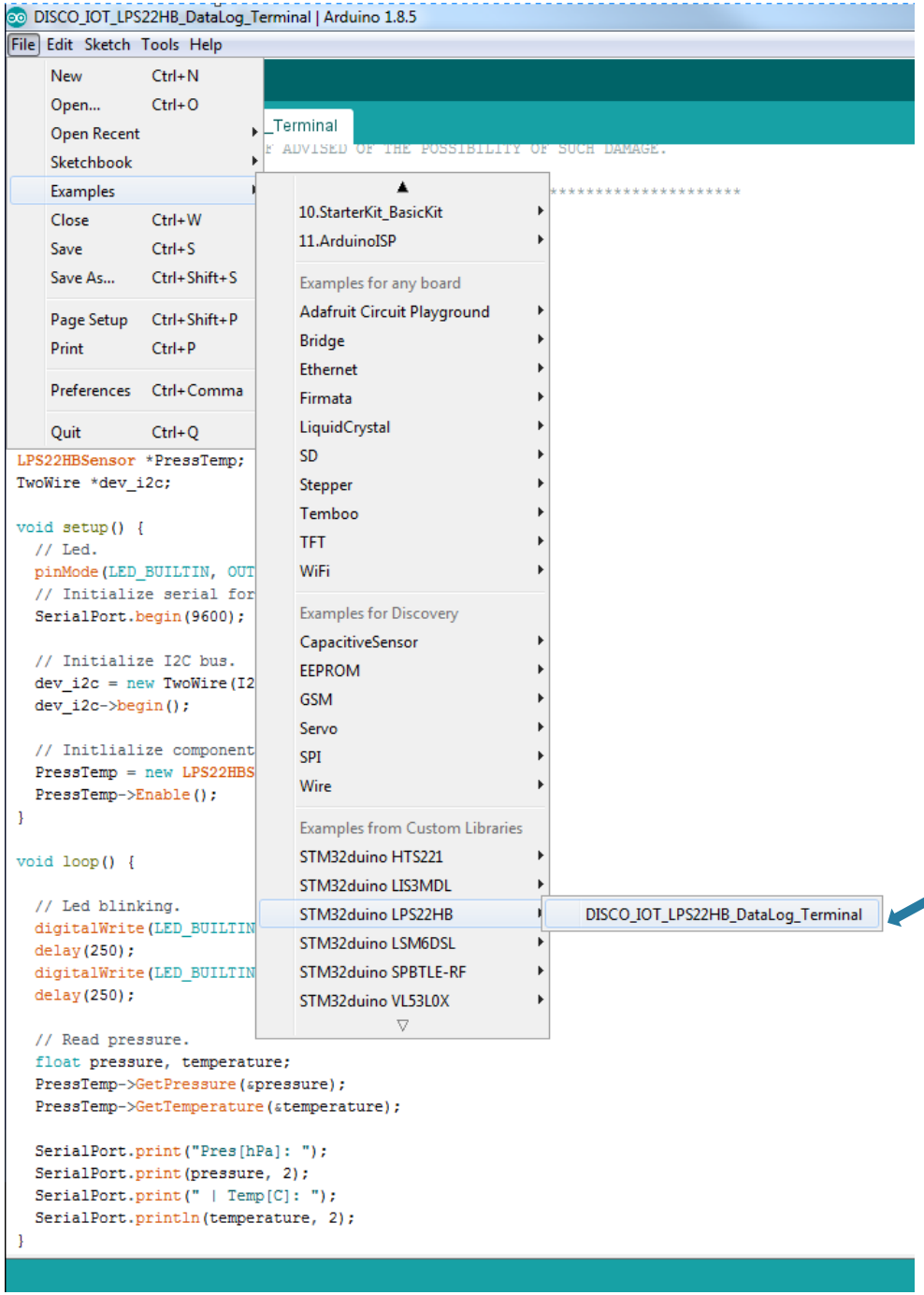
Barometer

- Objectif:
  - Lire les valeurs de la pression et de température du LPS22HB se trouvant sur la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer ces valeurs et les afficher sur le terminal

- Outils:
  - ARDUINO IDE et serial monitor
  - Librairie « STM32duino_ LPS22HB »
  - Exemple "DISCO_IOT_ LPS22HB_DataLog_Terminal "

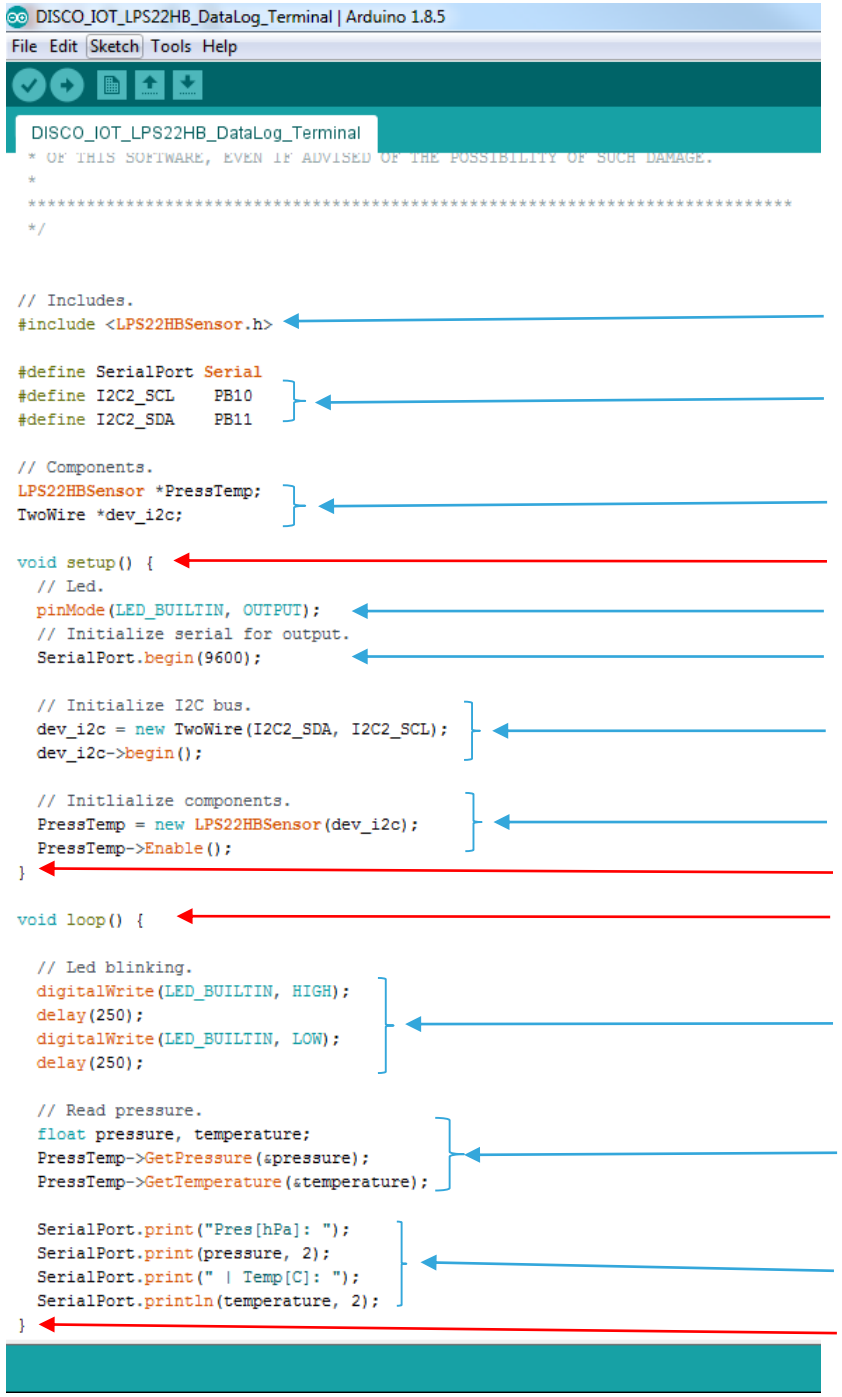
Barometer

- Télécharger la librairie « STM32duino_ LPS22HB »
- Ouvrir l'exemple « DISCO_IOT_ LPS22HB _DataLog_Terminal »



life.augmented

Barometer



Librairie des fonctions du LPS22HB

Définition des broches du bus I2C

Instantiation des classes LPS22HB et I2C

Début de la fonction d'initialisation

Initialisation du port en sortie pour la LED

Initialisation de l'UART a 9600 BAUD

Initialisation du bus I2C

Initialisation du LPS22HB

Fin de la fonction d'initialisation

Début de la fonction de la boucle principale

Clignotement de la LED

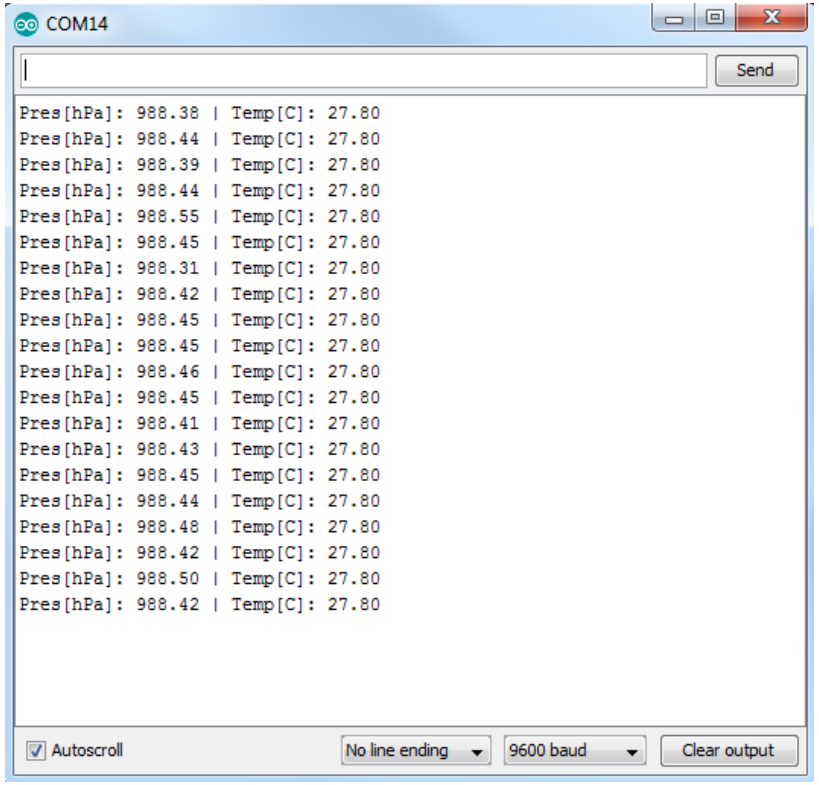
Lecture de la pression et de la temperature provenant du LPS22HB

Ecriture de la pression et de la temperature sur le terminal via l'UART

Fin de la fonction de la boucle principale

Barometer

- Ouvrir “serial monitor” pour espionner la lecture/ecriture

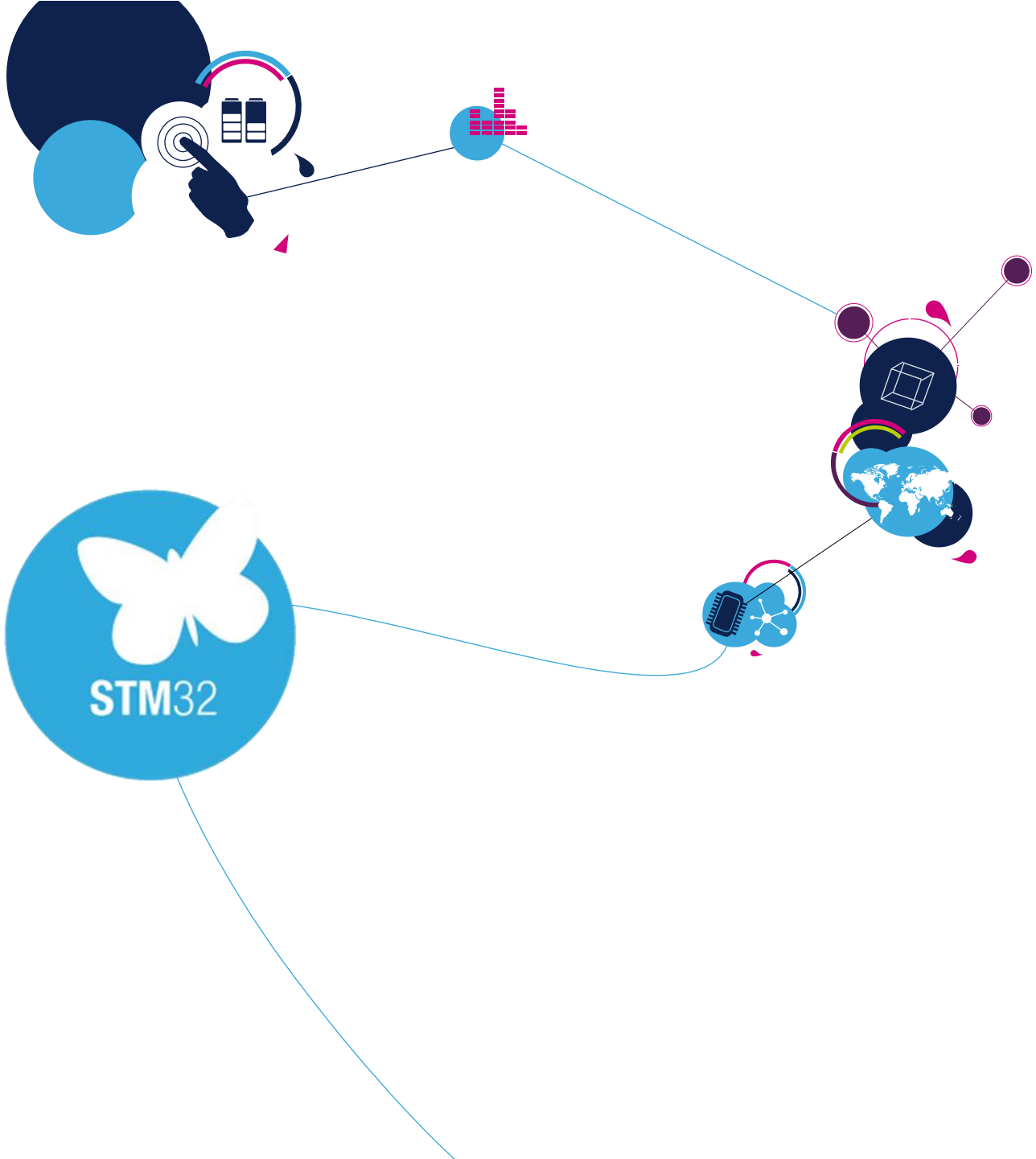


Checkpoint # 4
Barometer data

life.augmented

Hands On MEMs

LSM6DSL (3D Accelometer & Gyroscope)



life.augmented

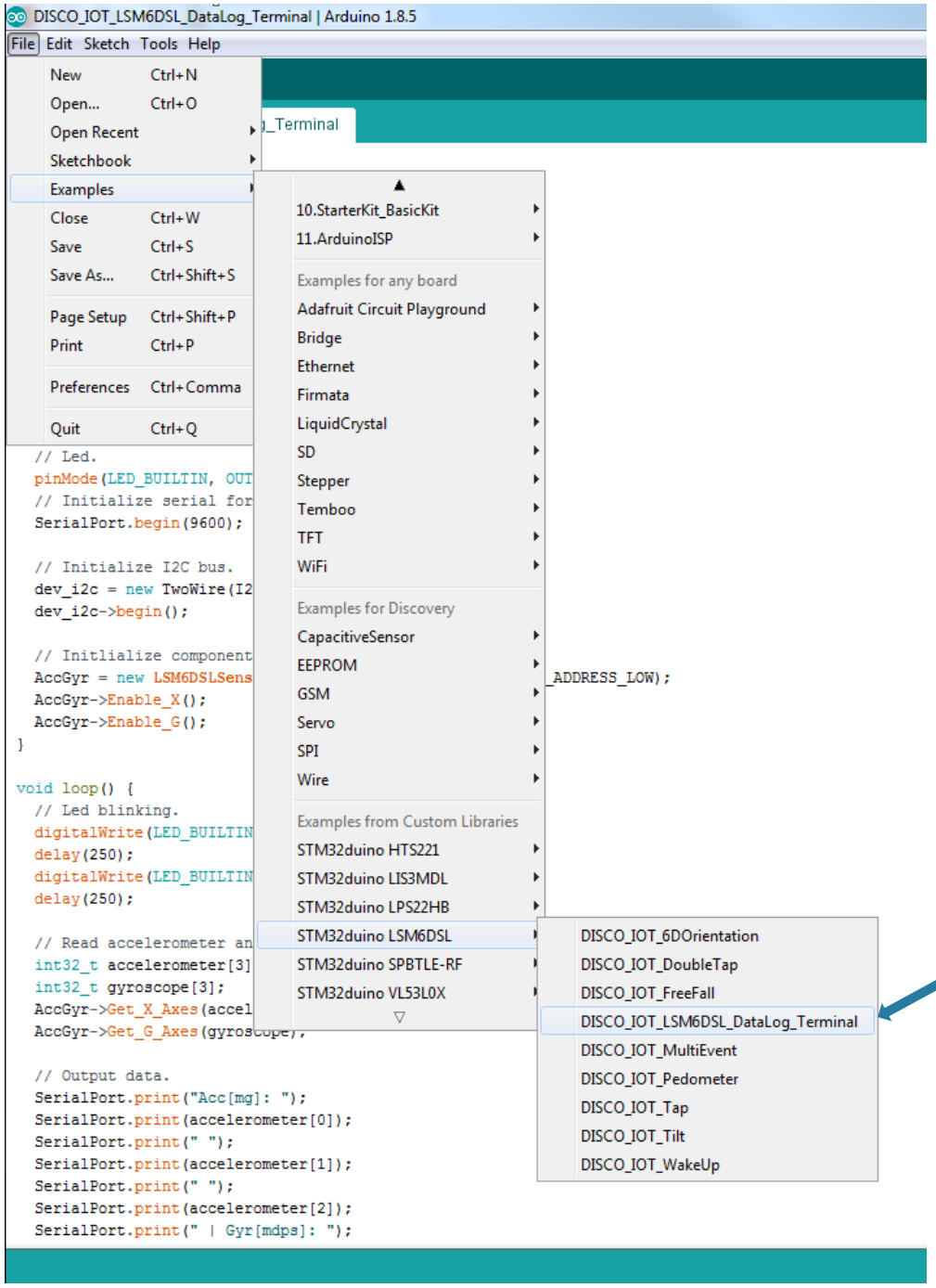
3D Accelometer & Gyroscope

- Objectif:
  - Lire les valeurs des 3 axes de l'accéléromètre et du gyroscope du LSM6DSL se trouvant sur la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer ces valeurs et les afficher sur le terminal

- Outils:
  - ARDUINO IDE et serial monitor
  - Librairie « STM32duino_ LSM6DSL »
  - Exemple "DISCO_IOT_ LSM6DSL_DataLog_Terminal "

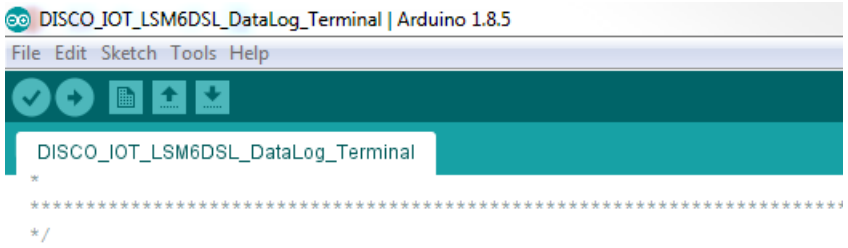
3D Accelometer & Gyroscope

- Télécharger la librairie « STM32duino_ LSM6DSL »
- Ouvrir l'exemple « DISCO_IOT_ LSM6DSL _DataLog_Terminal »



life.augmented

3D Accelometer & Gyroscope (1)




```
 *
 ******************************************************************************
 */


// Includes.
#include <LSM6DSLSensor.h>

#define SerialPort Serial
#define I2C2_SCL    PB10
#define I2C2_SDA    PB11

// Components.
LSM6DSLSensor *AccGyr;
TwoWire *dev_i2c;

void setup() {
  // Led.
  pinMode(LED_BUILTIN, OUTPUT);
  // Initialize serial for output.
  SerialPort.begin(9600);

  // Initialize I2C bus.
  dev_i2c = new TwoWire(I2C2_SDA, I2C2_SCL);
  dev_i2c->begin();

  // Initlialize components.
  AccGyr = new LSM6DSLSensor(dev_i2c, LSM6DSL_ACC_GYRO_I2C_ADDRESS_LOW);
  AccGyr->Enable_X();
  AccGyr->Enable_G();
}

void loop() {
  // Led blinking.
  digitalWrite(LED_BUILTIN, HIGH);
  delay(250);
  digitalWrite(LED_BUILTIN, LOW);
  delay(250);

  // Read accelerometer and gyroscope.
  int32_t accelerometer[3];
  int32_t gyroscope[3];
  AccGyr->Get_X_Axes(accelerometer);
  AccGyr->Get_G_Axes(gyroscope);
```

Librairie des fonctions du LSM6DSL

Définition des broches du bus I2C

Instantiation des classes LSM6DSL et I2C

Début de la fonction d'initialisation

Initialisation du port en sortie pour la LED

Initialisation de l'UART a 9600 BAUD

Initialisation du bus I2C

Initialisation du LSM6DSL

Fin de la fonction d'initialisation

Début de la fonction de la boucle principale

Clignotement de la LED

Lecture des 3 axes de l'accéléromètre et du gyroscope en provenance du LSM6DSL

3D Accelometer & Gyroscope (2)


```
// Output data.
SerialPort.print("Acc[mg]: ");
SerialPort.print(accelerometer[0]);
SerialPort.print(" ");
SerialPort.print(accelerometer[1]);
SerialPort.print(" ");
SerialPort.print(accelerometer[2]);
SerialPort.print(" | Gyr[mdps]: ");
SerialPort.print(gyroscope[0]);
SerialPort.print(" ");
SerialPort.print(gyroscope[1]);
SerialPort.print(" ");
SerialPort.println(gyroscope[2]);
}
```

Ecriture des valeurs des 3 axes de l'accéléromètre et du gyroscope sur le terminal via l'UART

Fin de la fonction de la boucle principale

3D Accelometer & Gyroscope

- Ouvrir “serial monitor” pour espionner la lecture/ecriture

COM14

Send

```
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -980 -2030 2100
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -980 -2030 2100
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -30 1033 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -30 1035 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -29 1033 | Gyr[mdps]: -910 -2030 2170
Acc[mg]: -1 -30 1032 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -980 -1960 2170
Acc[mg]: 3 -37 998 | Gyr[mdps]: -1680 -1540 2170
Acc[mg]: 0 -30 1035 | Gyr[mdps]: -980 -1960 2100
Acc[mg]: 0 -30 1034 | Gyr[mdps]: -910 -2030 2170
Acc[mg]: 0 -30 1034 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -30 1034 | Gyr[mdps]: -1050 -2030 2170
Acc[mg]: 0 -30 1033 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -29 1034 | Gyr[mdps]: -1050 -2030 2170
Acc[mg]: 0 -30 1035 | Gyr[mdps]: -980 -2030 2170
Acc[mg]: 0 -30 1035 | Gyr[mdps]: -980 -1960 2170
Acc[mg]: 0 -30 1036 | Gyr[mdps]: -980 -1960 2170
Acc[mg]: 0 -30 1035 | Gyr[mdps]: -980 -2030 217
```

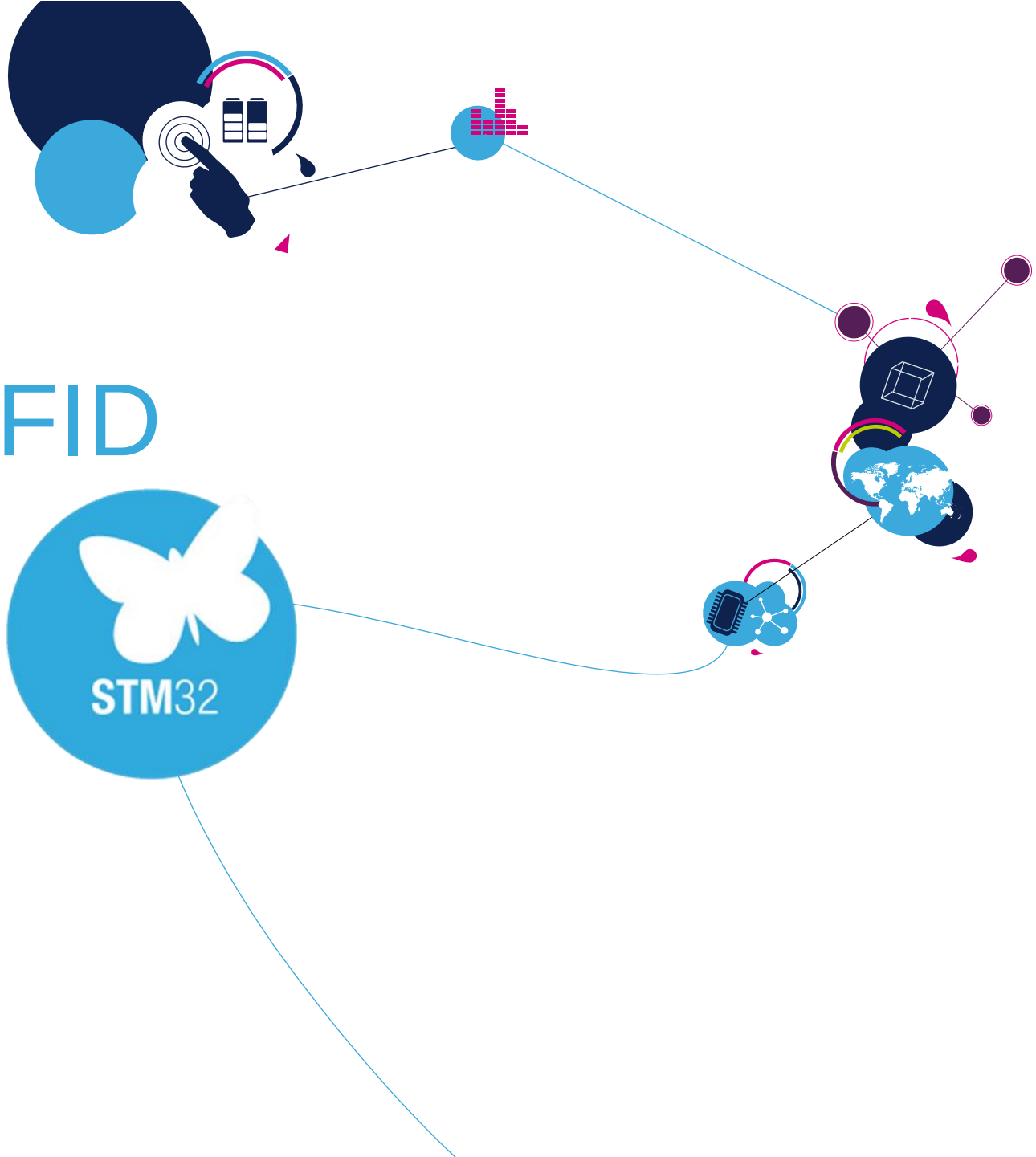
Autoscroll | No line ending | 9600 baud | Clear output

Checkpoint # 5
MEMs data

life.augmented

Hands On NFC/RFID

M24SR64-Y



life.augmented

NFC / RFID

- Objectif:
  - Lire et écrire l'URL d'un TAG NFC avec la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer la valeur du TAG lu
- Outils
  - Smartphone & Application « NFC tools »
  - ARDUINO IDE et serial monitor
  - Librairie « STM32duino_M24SR64-Y »
  - Exemple “ Write URI “

NFC / RFID

- Telecharger la librairie « STM32duino_M24SR64-Y »
- Ouvrir l'exemple «Write URI »

NFC / RFID

WriteURI

```
 * Write a Ndef URI message, wait and read the message from
 * the Nfc tag.
 *
 */
void setup() {
  const char uri_write[] = "stm.com";       // Uri to write in the tag
  char uri_read[255] = {'\0'};              // Uri read in the tag

  // Initialize serial for output.
  SerialPort.begin(9600);

  // Initialize I2C bus.
  dev_i2c.begin();

  // Intialize NFC module
  if(nfcTag.begin(NULL) == 0) {
    SerialPort.println("System Init done!");
  } else {
    SerialPort.println("System Init failed!");
    while(1);
  }

  delay(100);

  if(nfcTag.writeUri(uri_write) == false) {
    SerialPort.println("Write failed!");
    while(1);
  }

  delay(100);

  //read the txt message and print it
  nfcTag.readUri(uri_read);
  SerialPort.print("URI: ");
  SerialPort.println(uri_read);

  if(strcmp(uri_write, uri_read) == 0) {
    SerialPort.println("Successfully written and read!");
  } else {
    SerialPort.println("Read failed!");
  }
}

void loop() {
  //empty loop

  char uri_read_2[255] = {'\0'};            // Uri read in the tag
  //read the txt message and print it
  nfcTag.readUri(uri_read_2);
  SerialPort.print("URI: ");
  SerialPort.println(uri_read_2);
  delay(1000);
}
```

Initialisation de l'UART a 9600 BAUD

Initialisation du modul NFC

Ecriture du TAG

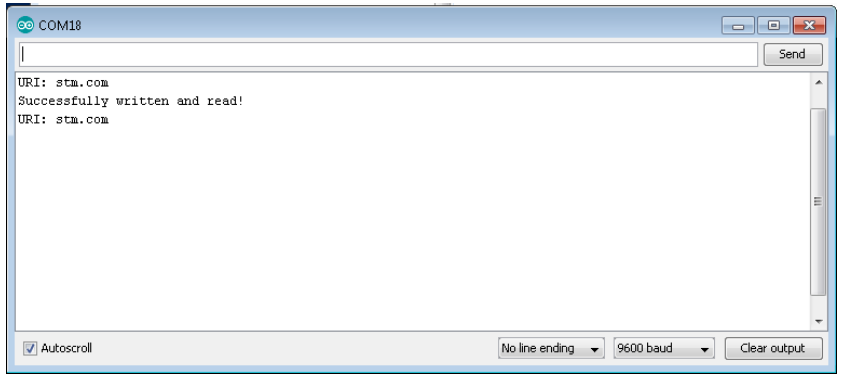
Lecture du TAG

Comparaison du TAG ecrit et lu

Lecture du TAG dans une boucle perpétuelle avec 1s de temporisation

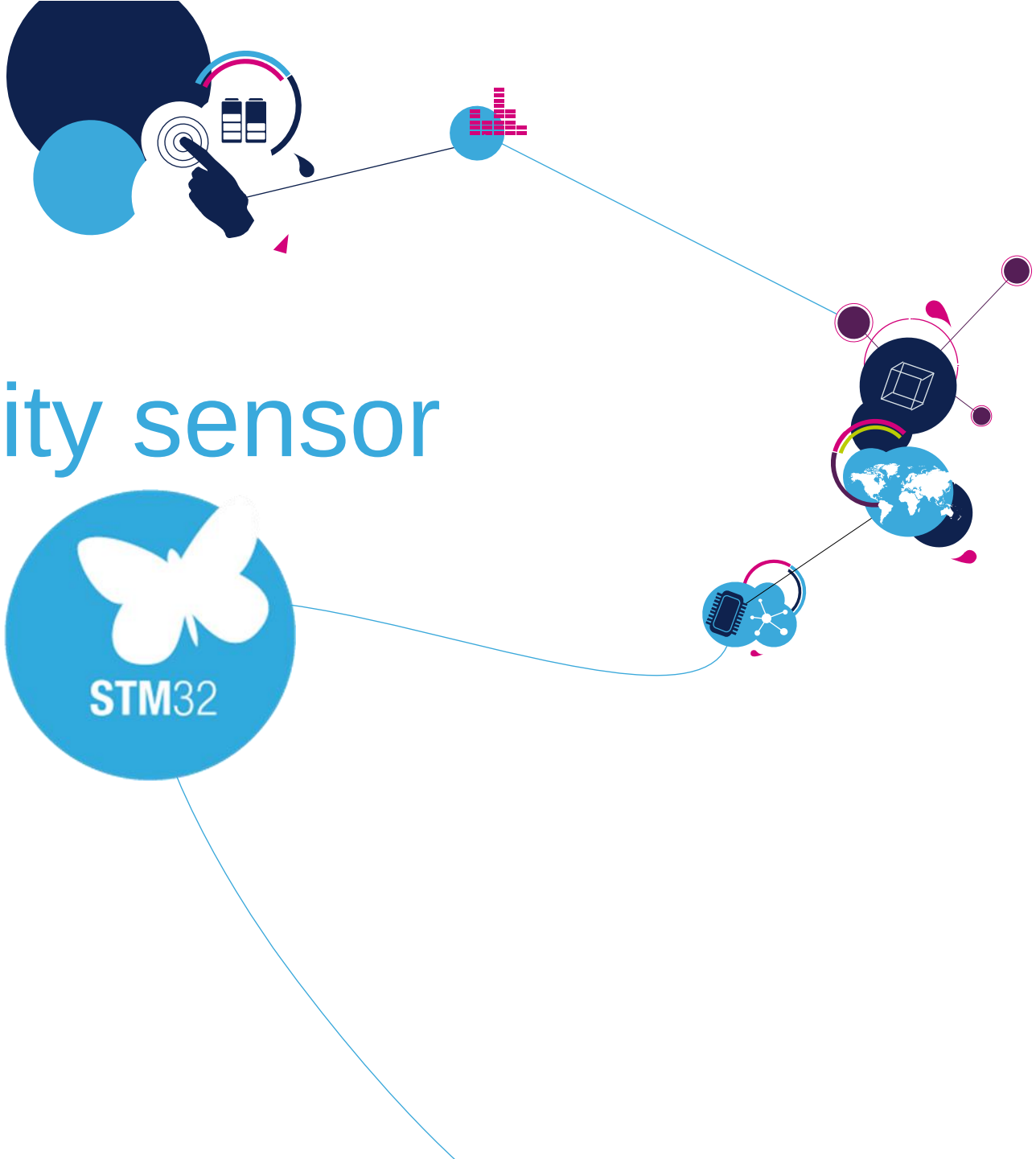
NFC / RFID

- Ouvrir “serial monitor” pour espionner la lecture/ecriture



Checkpoint # 6
NFC write/read

life.augmented



Hands On Proximity sensor

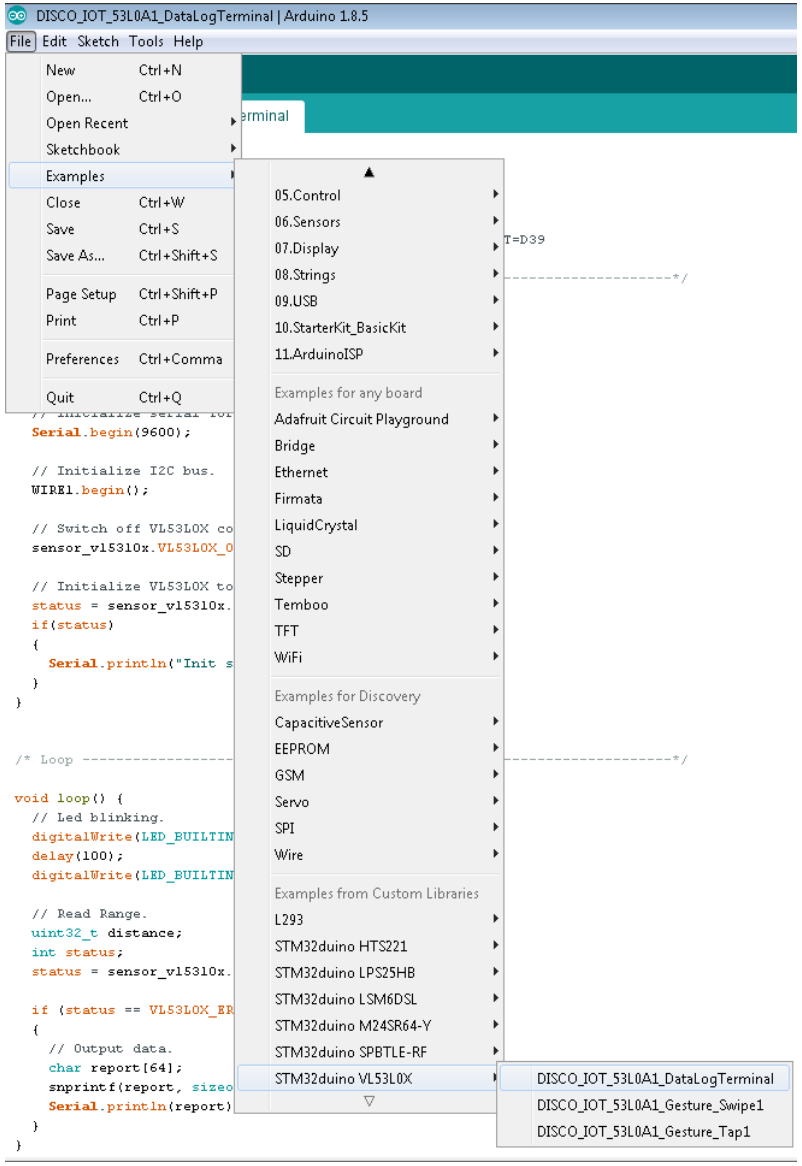
VL53L0X (Time-Of-Flight)

life.augmented

Proximity sensor

- Objectif:
  - Lire la distance entre le capteur TOF et un objet avec la carte Discovery B-L475E-IOT01A IoT node
  - Utiliser l'UART pour récupérer la distance lu
- Outils
  - ARDUINO IDE et serial monitor ou serial plotter
  - Librairie « STM32duino_VL53L0X »
  - Exemple “ DISCO_IOT_53L0A1_DataLogterminal “

Proximity sensor



life.augmented

Proximity sensor

DISCO_IOT_53L0A1_DataLogTerminal §

```
#include <Wire.h>
#include <vl53l0x_class.h>

// Create components.
TwoWire WIRE1(D34, D33);  //SDA=D34 & SCL=D33
VL53L0X sensor_vl53l0x(&WIRE1, D38, D39); //XSHUT=D38 & INT=D39

/* Setup --------------------------------------------------------------------*/

void setup() {
  int status;
  // Led.
  pinMode(LED_BUILTIN, OUTPUT);

  // Initialize serial for output.
  Serial.begin(9600);

  // Initialize I2C bus.
  WIRE1.begin();

  // Switch off VL53L0X component.
  sensor_vl53l0x.VL53L0X_Off();

  // Initialize VL53L0X top component.
  status = sensor_vl53l0x.InitSensor(0x10);
  if(status)
  {
    Serial.println("Init sensor_vl53l0x failed...");
  }
}

/* Loop ---------------------------------------------------------------------*/

void loop() {
  // Led blinking.
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);

  // Read Range.
  uint32_t distance;
  int status;
  status = sensor_vl53l0x.GetDistance(&distance);

  if (status == VL53L0X_ERROR_NONE)
  {
    // Output data.
    char report[64];
    snprintf(report, sizeof(report), "| Distance [mm]: %ld |", distance);
    Serial.println(report);
  }
}
```

Librairie des fonctions du vl5310x

Definition du capteur

Début de la fonction d'initialisation

Init de la Led

Initialisation de l'UART a 9600 BAUD

Initialisation du vl5310x

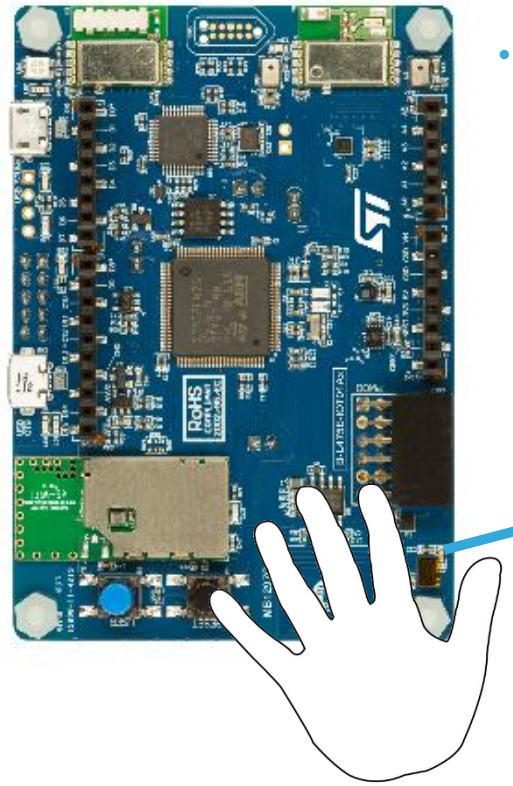
Début de la fonction de la boucle principale

Clignotement de la LED

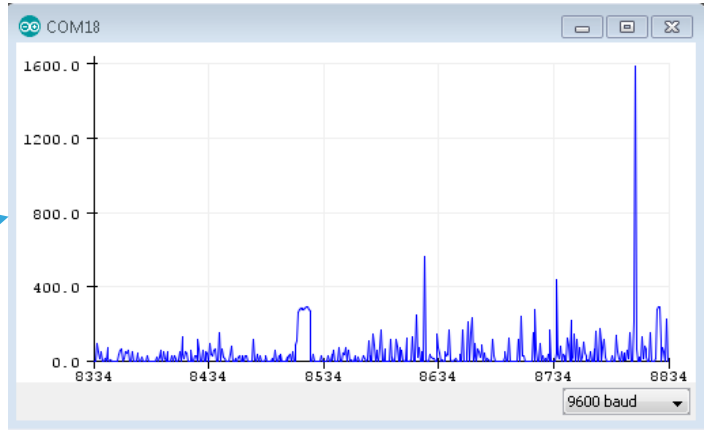
Lecture de la distance

Fin de la fonction de la boucle principale

Proximity sensor

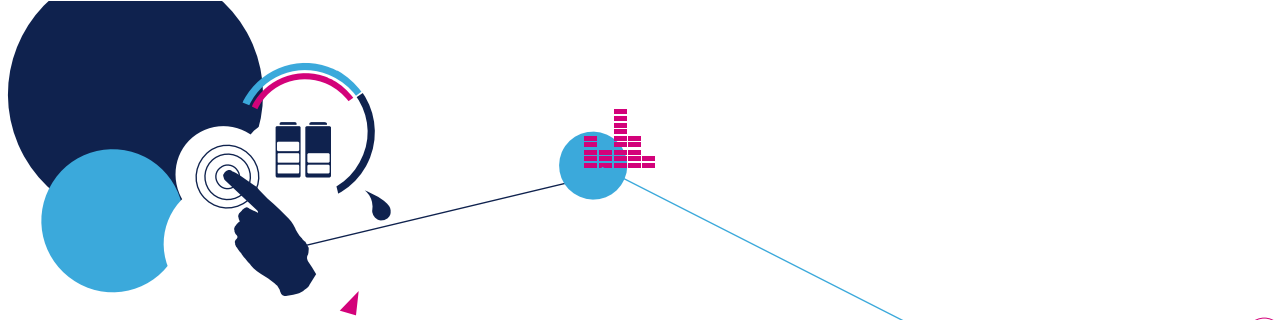


- Ouvrir “serial monitor” pour espionner la lecture/ecriture



Checkpoint # 7
TimeOfFlight data

life.augmented



Hands On Bluetooth Low Energy

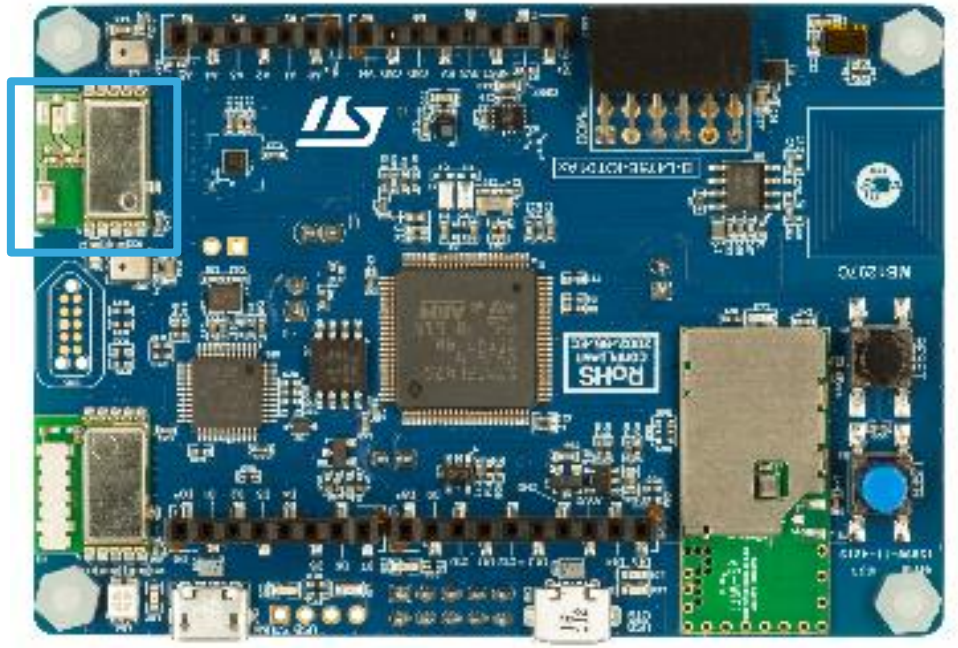
SPBTLE-RF

life.augmented

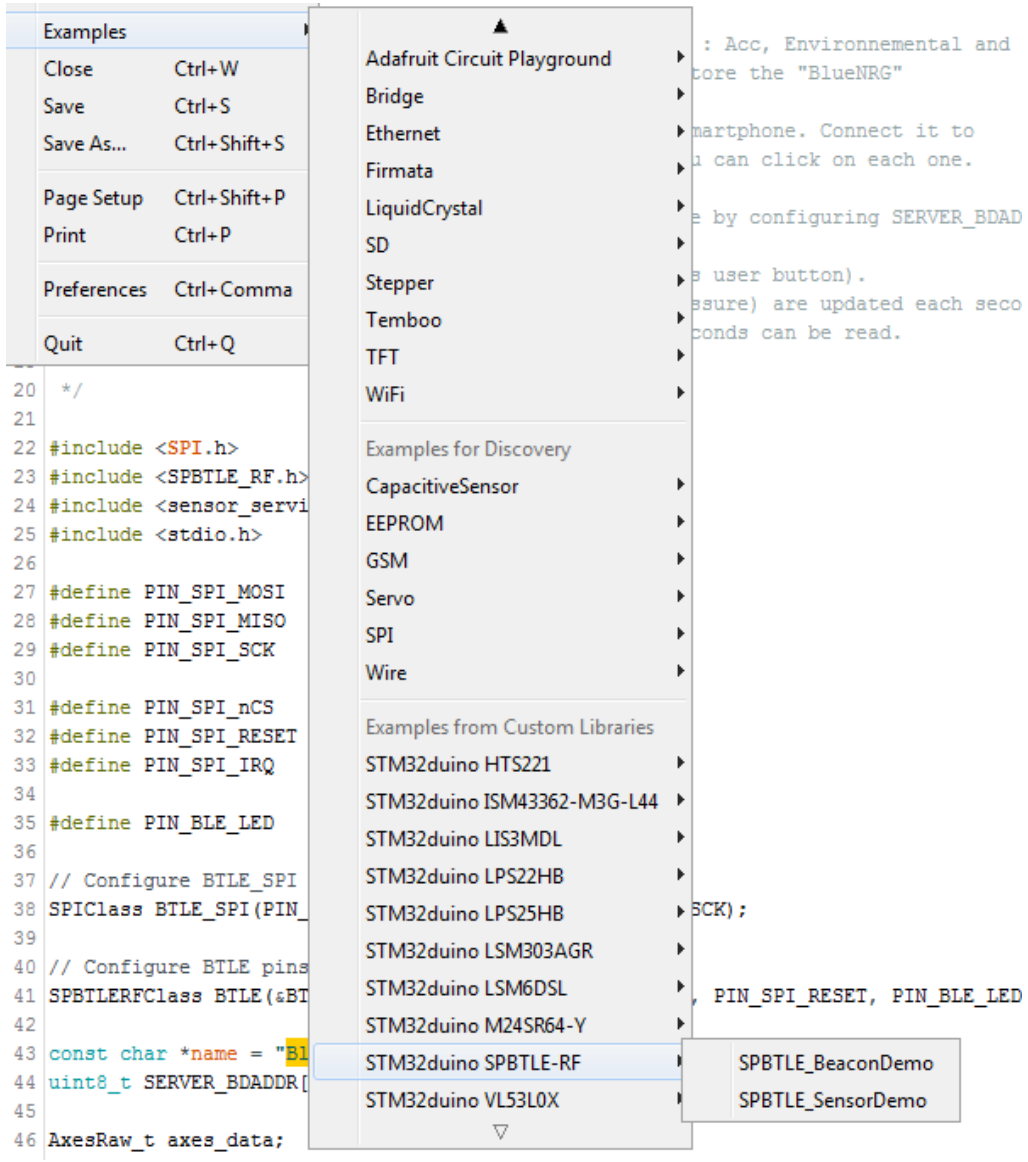
Bluetooth Low Energy

- Objectif:
  - Connecter le Bluetooth Low Energy de la carte IoT node avec son smartphone
  - Ajouter au projet SensorDemo les donnees recuperes des capteurs

- Outils
  - ARDUINO IDE et serial monitor
  - Librairie « SPBTLE-RF »
  - Exemple “ SPBTLE_SensorDemo “
  - Application BLUE NRG



Bluetooth Low Energy



life.augmented

Bluetooth Low Energy (1)


```
22 #include <SPI.h>
23 #include <SPBTLE_RF.h>
24 #include <sensor_service.h>
25 #include <stdio.h>
26
27 #define PIN_SPI_MOSI    (PC12)
28 #define PIN_SPI_MISO    (PC11)
29 #define PIN_SPI_SCK     (PC10)
30
31 #define PIN_SPI_nCS     (PD13)
32 #define PIN_SPI_RESET   (PA8)
33 #define PIN_SPI_IRQ     (PE6)
34
35 #define PIN_BLE_LED     (LED4)
36
37 // Configure BTLE_SPI
38 SPIClass BTLE_SPI(PIN_SPI_MOSI, PIN_SPI_MISO, PIN_SPI_SCK);
39
40 // Configure BTLE pins
41 SPBTLERFClass BTLE(&BTLE_SPI, PIN_SPI_nCS, PIN_SPI_IRQ, PIN_SPI_RESET, PIN_BLE_LED);
42
43 const char *name = "BlueNRJ";
44 uint8_t SERVER_BDADDR[] = {0x15, 0x34, 0x00, 0xE1, 0x80, 0x03};
45
46 AxesRaw_t axes_data;
47 uint32_t previousSecond = 0;
48
49 void setup() {
50   int ret;
51
52   Serial.begin(9600);
53   if(BTLE.begin() == SPBTLERF_ERROR)
54   {
55     Serial.println("Bluetooth module configuration error!");
56     while(1);
57   }
```

Definition of STLBLE services

SPI's GPIO definition

LED's GPIO definition

BLE SPI Initilization

BLE service name
BLE MAC address (Change first byte)

Raw axes data of x,y,z

Start of Setup function

Uart initialization

BLE initialization

Bluetooth Low Energy (2)


```
59  if(SensorService.begin(name, SERVER_BDADDR))
60  {
61    Serial.println("Sensor service configuration error!");
62    while(1);
63  }
64
65  /* Configure the User Button in GPIO Mode */
66  pinMode(USER_BTN, INPUT);
67
68  ret = SensorService.Add_Acc_Service();
69
70  if(ret == BLE_STATUS_SUCCESS)
71    Serial.println("Acc service added successfully.");
72  else
73    Serial.println("Error while adding Acc service.");
74
75  ret = SensorService.Add_Environmental_Sensor_Service();
76
77  if(ret == BLE_STATUS_SUCCESS)
78    Serial.println("Environmental Sensor service added successfully.");
79  else
80    Serial.println("Error while adding Environmental Sensor service.");
81
82  randomSeed(analogRead(A0));
83
84  /* Instantiate Timer Service with two characteristics:
85   * - seconds characteristic (Readable only)
86   * - minutes characteristics (Readable and Notifiable )
87   */
88  ret = SensorService.Add_Time_Service();
89
90  if(ret == BLE_STATUS_SUCCESS)
91    Serial.println("Time service added successfully.");
92  else
93    Serial.println("Error while adding Time service.");
94 }
95
```

Start of the sensor GATT service

Add accelerometer service using a vendor specific profile

Add environmental service using a vendor specific profile (Temperature, Pressure & Humidity)

Initialize randomization function

Add time service using a vendor specific profile

End of setup function

life.augmented

Bluetooth Low Energy (3)


```
96  void loop() {
97    BTLE.update();
98
99    if(SensorService.isConnected() == TRUE)
100   {
101     //Update accelerometer values
102     User_Process(&axes_data);
103
104     //Update time
105     SensorService.Update_Time_Characteristics();
106
107     if((millis() - previousSecond) >= 1000)
108     {
109       //Update environnemental data
110       //Data are set with random values but can be replace with data from sensors.
111       previousSecond = millis();
112       SensorService.Temp_Update(random(-100,400));
113       SensorService.Press_Update(random(95000,105000));
114       SensorService.Humidity_Update(random(0,100));
115     }
116   }
117   else
118   {
119     //Keep the Bluetooth module in discoverable mode
120     SensorService.setConnectable();
121   }
122 }
123
124 /**
125  * @brief  Process user input (i.e. pressing the USER button on Nucleo board)
126  *         and send the updated acceleration data to the remote client.
127  *
128  * @param  AxesRaw_t* p_axes
129  * @retval None
130  */
131 void User_Process(AxesRaw_t* p_axes)
132 {
133   /* Check if the user has pushed the button */
134   if(digitalRead(USER_BTN) == RESET)
135   {
136     while (digitalRead(USER_BTN) == RESET);
```

Main Loop

Process HCI BLE packets

Check if BLE is connected (Bounded)

Update accelerometer values

Update time service

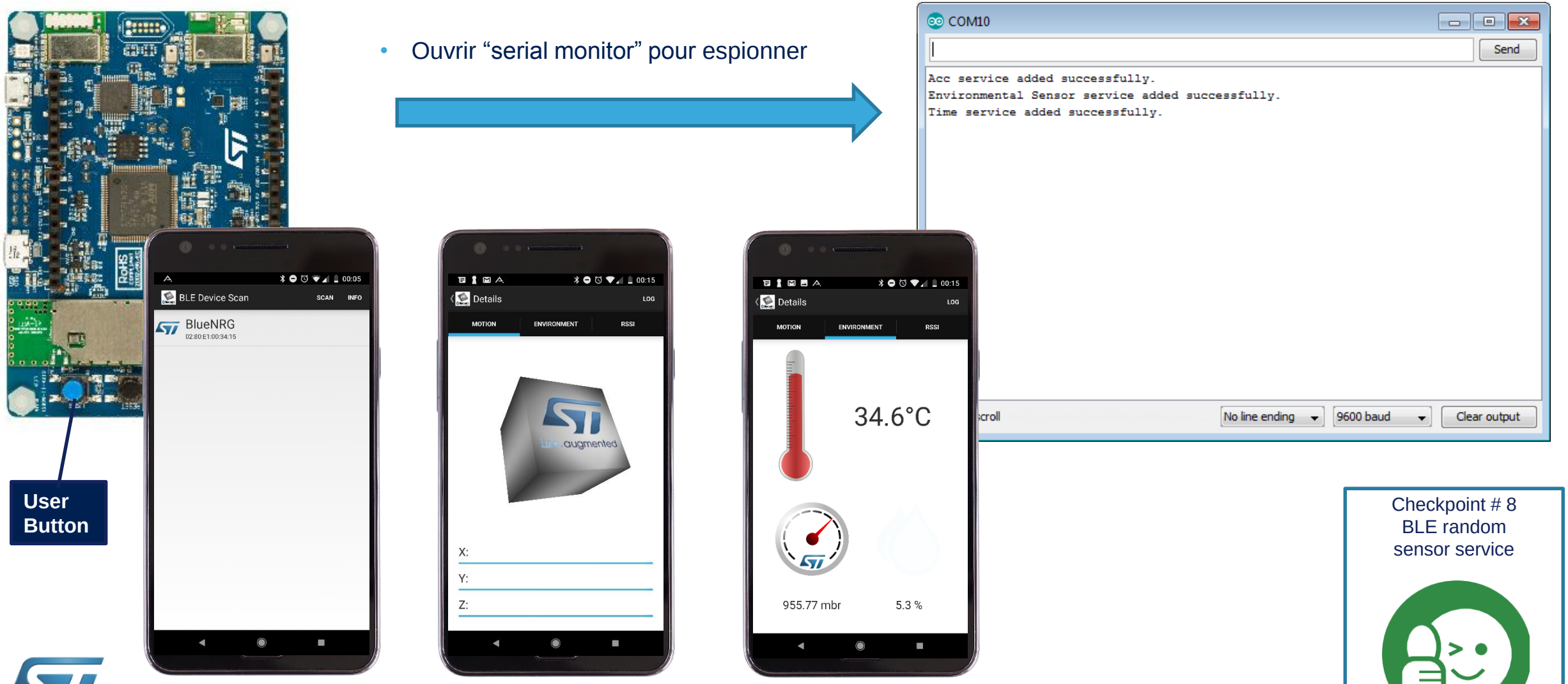
Update environmental service

Bluethooth is discoverable

Function called when button pressed to increase axes

Bluetooth Low Energy

- Ouvrir “serial monitor” pour espionner



Checkpoint # 8
BLE random
sensor service

life.augmented

Bluetooth Low Energy

- Ajouter au projet SensorDemo les données récupérées des capteurs

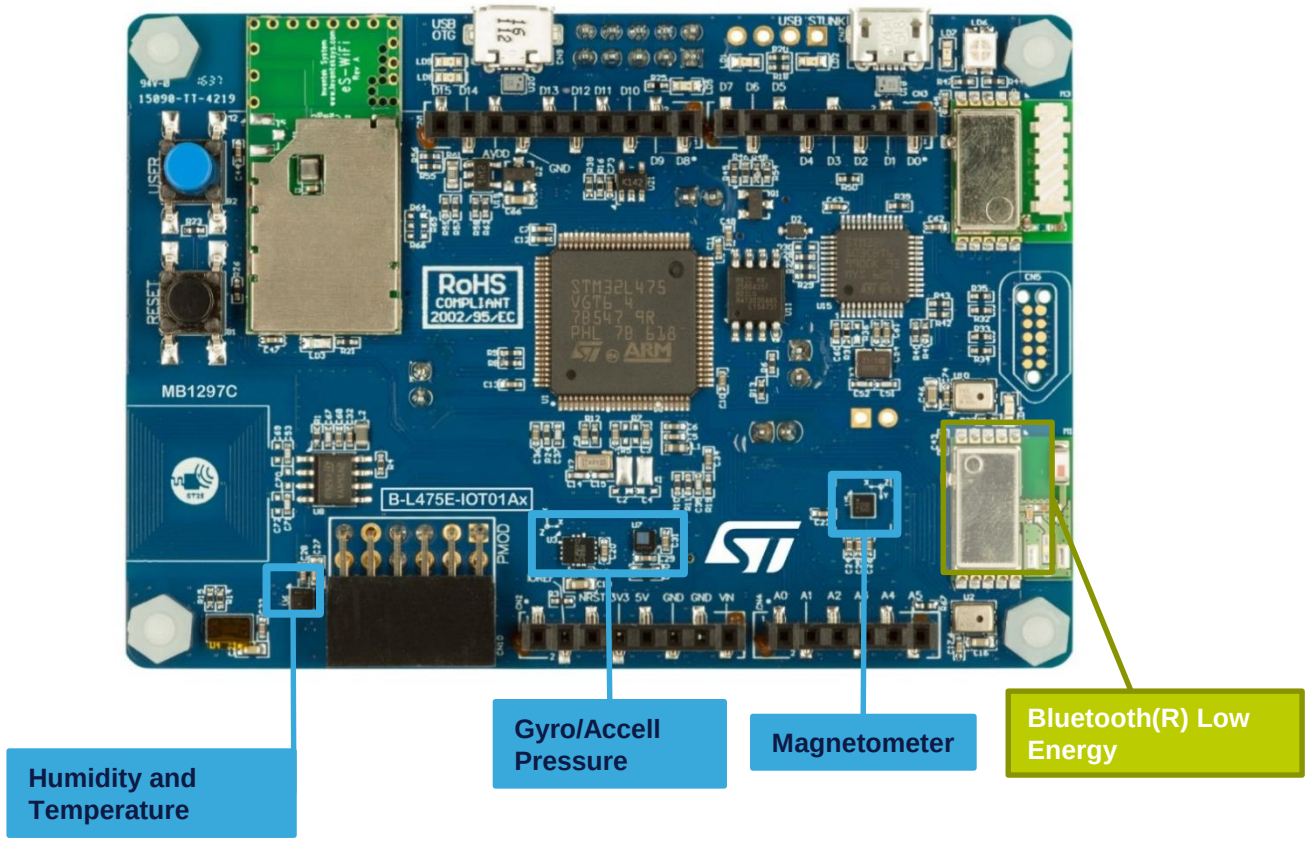
- Libraries a ajouter :
  - Température, Pression, Hygrométrie

    #include <HTS221Sensor.h>

    #include <LPS22HBSensor.h>
  - Gyroscope, Accéléromètre

    #include <LSM6DSLSensor.h>

A vos claviers… ;-)



life.augmented

Bluetooth Low Energy (1)


```
27 #include <HTS221Sensor.h>
28 #include <LPS22HBSensor.h>
29 #include <LSM6DSLSensor.h>
```

Add sensors defines

```
52 /* i2c sensors */
53 TwoWire *dev_i2c;
54 #define I2C2_SCL     PB10
55 #define I2C2_SDA     PB11
56 #define INT1         PD11
57
58 HTS221Sensor  *HumTemp;
59 LPS22HBSensor *PressTemp;
60 LSM6DSLSensor *AccGyr;
61
62 void setup() {
63   int ret;
64
65   Serial.begin(9600);
66   // Initialize I2C bus.
67   dev_i2c = new TwoWire(I2C2_SDA, I2C2_SCL);
68   dev_i2c->begin();
69
70   // Initlialize components.
71   HumTemp = new HTS221Sensor (dev_i2c);
72   HumTemp->Enable();
73
74   PressTemp = new LPS22HBSensor(dev_i2c);
75   PressTemp->Enable();
76
77   // Initlialize Components.
78   AccGyr = new LSM6DSLSensor(dev_i2c, LSM6DSL_ACC_GYRO_I2C_ADDRESS_LOW);
79   AccGyr->Enable_X();
```

I2CGPIO definition

Global variables

Init I2C

Initialization of the sensors

life.augmented

Bluetooth Low Energy (2)


```
122 void loop() {
123   BTLE.update();
```

```
134     if((millis() - previousSecond) >= 1000)
135     {
136       float humidity, temperature;
137       float pressure, temperature_lps22hb;
138
139       HumTemp->GetHumidity(&humidity);
140       HumTemp->GetTemperature(&temperature);
141
142       PressTemp->GetPressure(&pressure);
143       PressTemp->GetTemperature(&temperature_lps22hb);
144
145       //Update environnemental data
146       SensorService.Temp_Update(temperature*10);
147       SensorService.Press_Update(pressure*100);
148       SensorService.Humidity_Update(humidity*10);
149     }
```

Get Humidity and Temperature from sensor

Get Pressure and Temperature from sensor

Update values from sensors

```
164 void User_Process(AxesRaw_t* p_axes)
165 {
166   /* Check if the user has pushed the button */
167     int32_t accelerometer[3];
168     static int32_t accelerometer_init[3]={0};
169     AccGyr->Get_X_Axes(accelerometer);
170     if(accelerometer_init[0] == 0){
171       *accelerometer_init = *accelerometer;
172     }
173
174     p_axes->AXIS_X += (accelerometer[0]-accelerometer_init[0]);
175     p_axes->AXIS_X /= 2;
176     p_axes->AXIS_Y += (accelerometer[1]-accelerometer_init[1]);
177     p_axes->AXIS_Y /= 2;
178     p_axes->AXIS_Z += (accelerometer[2]-accelerometer_init[2]);
179     p_axes->AXIS_Z /= 2;
180     SensorService.Acc_Update(p_axes);
181     delay(50);
182 }
```

Compute new data from accelerometer
Update service with new axes values

life.augmented

Thank you

life.augmented

www.st.com/stm32

Add printf function (1)

- From Very lean Serial.printf() (www.stm32duino.com forum)

  Under directory ***C:/…/arduino-1.8.5/hardware/arduino/avr/cores/Arduino***

  Following code to add :

  - **Print.h** :
    - Line 88: int printf(const char * format, ...);
  - **Print.cpp** :
    - Line 27 add :

```
#include <stdarg.h>
typedef void (*pfn_outputchar)(char c, void* p);
extern "C"{
int _print_format (pfn_outputchar pfn, void* pvoid, const char *format, va_list ap);
}
```

Add printf function (2)

- **Print.cpp** Line 273 add :

```
static void put_char_to_string (char c, void* p)
{
  char **buf = (char **)p;
  *(*buf)++ = c;
}

static int vsprintf (char *buf, const char *format, va_list ap)
{
  int i;
  i = _print_format (put_char_to_string, &buf, format, ap);
  *buf = 0;
  return i;
}

int Print::printf (const char *format, ...)
{
    va_list arg;
    va_start(arg, format);
    char temp[64];
    char* buffer = temp;
    size_t len = vsprintf(temp, format, arg);
    va_end(arg);
    len = write((const uint8_t*) buffer, len);
    return len;
}
```

Add printf function (2)

- Print.cpp Line 273 add :

```
static void put_char_to_string (char c, void* p)
{
  char **buf = (char **)p;
  *(*buf)++ = c;
}

static int vsprintf (char *buf, const char *format, va_list ap)
{
  int i;
  i = _print_format (put_char_to_string, &buf, format, ap);
  *buf = 0;
  return i;
}

int Print::printf (const char *format, ...)
{
    va_list arg;
    va_start(arg, format);
    char temp[64];
    char* buffer = temp;
    size_t len = vsprintf(temp, format, arg);
    va_end(arg);
    len = write((const uint8_t*) buffer, len);
    return len;
}
```