

SENSORS &
FIELD TRANSMITTERS



HUMAN MACHINE
INTERFACE (HMI)



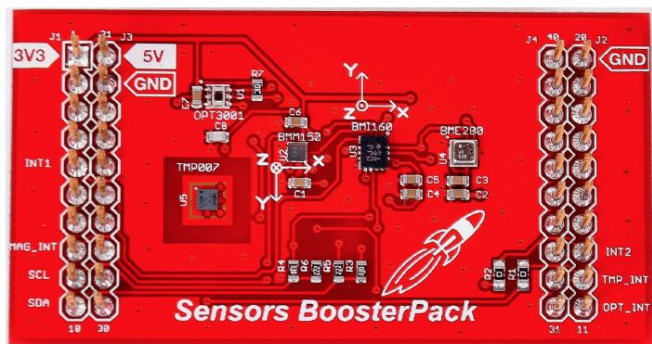
INDUSTRIAL
MOTOR DRIVE



INDUSTRIAL
COMMUNICATION

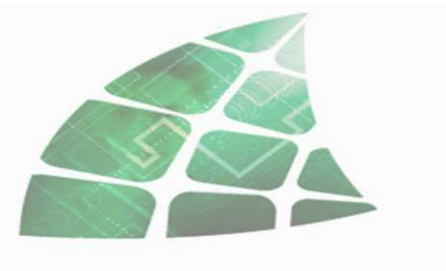


PROGRAMMABLE
LOGIC CONTROL (PLC)



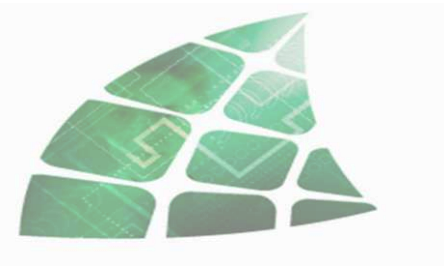
Tema 8. Manejo del Sensors Boosterpack

4º Grado de Ingeniería en Electrónica, Robótica y Mecatrónica
Andalucía Tech



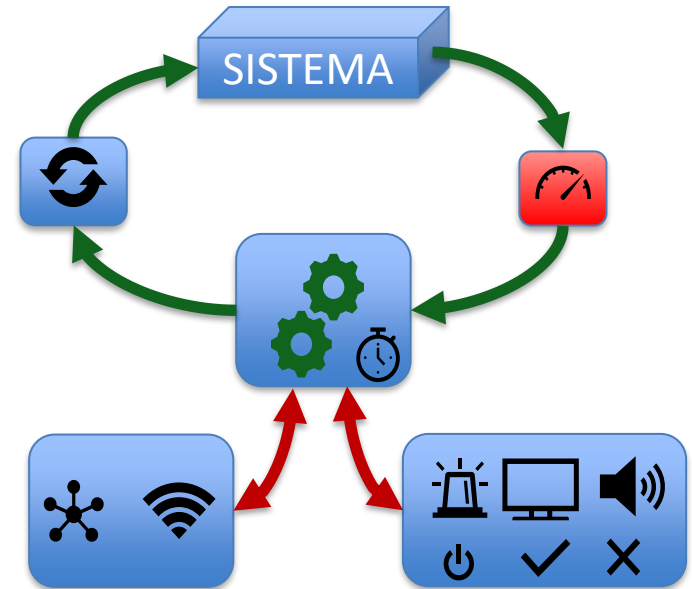
Índice

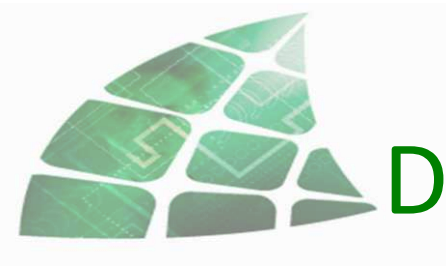
- Introducción
- Descripción física Sensors Boosterpack
- Librería Sensorlib2





Introducción

- Esquema general:
- Necesidad de medir
- Sensores digitales
 - Precio, optimización



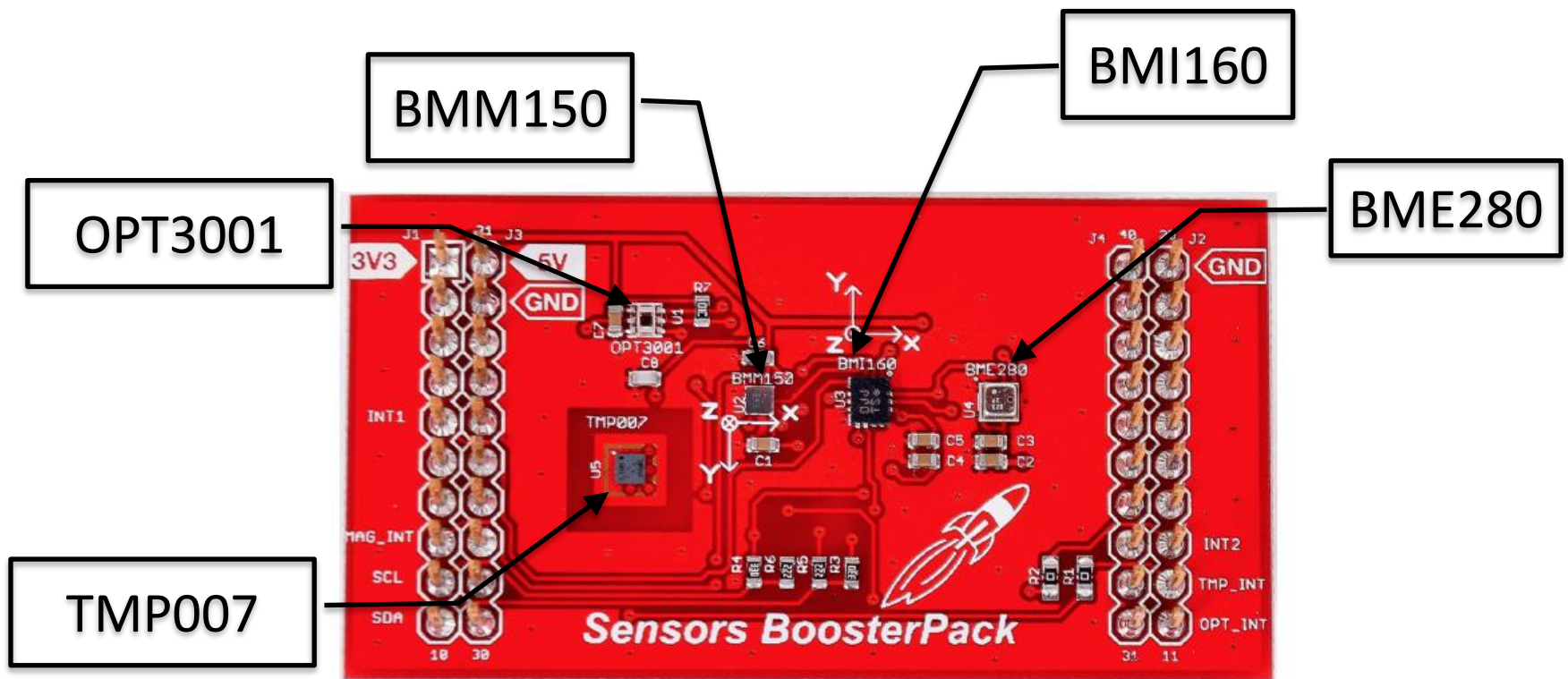


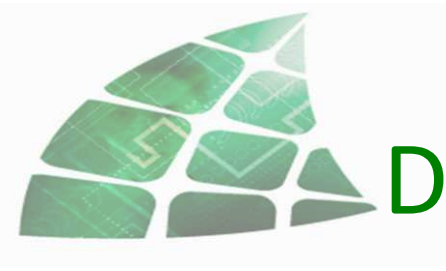
Descripción Sensors Boosterpack

- Segunda versión del SensorHub.
 - Eliminado interfaz RF
- Sensores homogeneizados (TI y BOSCH):
 -  {
 - TMP007: sensor de Temperatura (OJO: obsoleto)
 - OPT3001: Sensor de luz
 -  {
 - BME280: sensor ambiental (T, P, H)
 - BMI160: acelerómetro 3D y giróscopo 3D
 - BMM130: magnetómetro 3D

Descripción Sensors Boosterpack

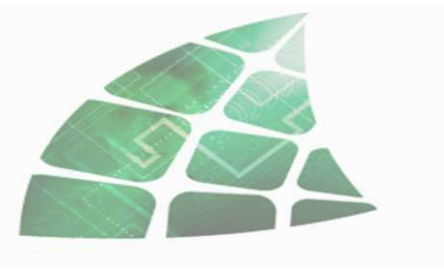
- Posición de los sensores:





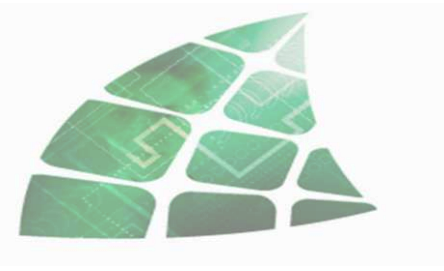
Descripción Sensors Boosterpack

- No hay una librería como tal para el TIVA
- Ejemplos de manejo para MSP432
- Fusión de código de TI y código de BOSH
- Manejo básico por polling, no por interrupciones
 - Más sencillo de manejar
 - Menos eficiente en tiempo y consumo



SENSORLIB2

- Librería recompilada de los ejemplos de TI
- Manejo simple para TIVA
- Ficheros necesarios:
 - Driver HAL:
 - HAL_I2C.C (Código fuente capa HAL)
 - HAL_I2C.h (Cabecera y definiciones)
 - Librería:
 - Sensorlib2.h (unión de varios ficheros de cabecera)
 - sensorlib2.lib (librería compilada)



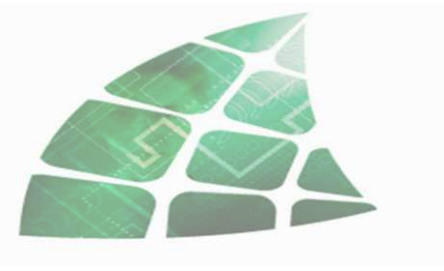
Funciones de capa HAL

- Comprueba, configura, lee o escribe

```
void Conf_Boosterpack(int pos, int RELOJ);  
uint8_t Test_I2C_Dir(uint8_t DIR);  
uint8_t Detecta_BP(int pos);
```

```
void I2C_init(void);  
int I2C_read16(unsigned int slaveAdr, unsigned char);  
void I2C_write16(unsigned char pointer, unsigned int writeByte);  
void I2C_setslave(unsigned int slaveAdr);
```

```
bool readI2C(uint8_t dev_addr, uint8_t reg_addr, uint8_t *reg_data, uint8_t cnt);  
bool writeI2C(char dev_addr, char reg_addr, char *reg_data, unsigned int cnt);
```

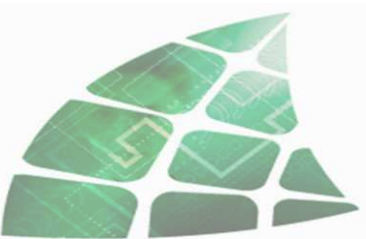



Funciones de inicialización

- Encienden y dan algunos valores iniciales:

```
void OPT3001_init(void);    // opt3001.h Inicialización opt3001

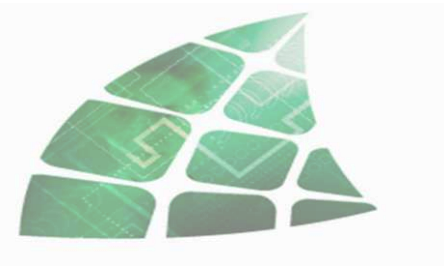
bool sensorTmp007Init(void);    // tmp007.h inicialización tmp007
bool sensorTmp007Enable(bool enable); // encender/apagar tmp007
s32 bme280_data_readout_template(void); // bme280.h inicialización bme280
    // modo del BME: BME280_SLEEP_MODE, BME280_FORCED_MODE, BME280_NORMAL_MODE
s8 bme280_set_power_mode(u8 v_power_mode_u8);
s8 bmi160_initialize_sensor(void); // bmi160.h inicialización bmi160
s8 bmi160_config_running_mode(u8 v_running_mode_u8); // Modo del bmi160:
/* STANDARD_UI_9DOF_FIFO STANDARD_UI_IMU_FIFO STANDARD_UI_IMU
STANDARD_UI_ADVANCEPOWERSAVE ACCEL_PEDOMETER APPLICATION_HEAD_TRACKING
APPLICATION_NAVIGATION APPLICATION_REMOTE_CONTROL APPLICATION_INDOOR_NAVIGATION */
```



Funciones de manejo básico

- Leer los valores de los sensores:
- Casi todas piden punteros a variables

```
float OPT3001_getLux(void);
bool sensorTmp007Read(int16_t *rawTemp, int16_t *rawObjTemp);
void sensorTmp007Convert(int16_t rawTemp, int16_t rawObjTemp, float *tObj, float *tAmb);
s8 bme280_read_pressure_temperature_humidity(u32 *presion, s32 *temp, u32 *hum);
// OJO: Devuelve P*100, T*100, H*1000
s8 bmi160_bmm150_mag_compensate_xyz( struct bmi160_mag_xyz_s32_t *mag_comp_xyz);
s8 bmi160_read_gyro_xyz(struct bmi160_gyro_t *gyro);
s8 bmi160_read_accel_xyz(struct bmi160_accel_t *accel);
```



Ejemplo 8

- Inicialización y uso de todos los sensores
- Se leen los DevID para comprobar el funcionamiento
- Se saca por consola el valor instantáneo
- Se mide (con un timer de 100us) lo que tarda en hacer todas las peticiones (menos de 2ms)