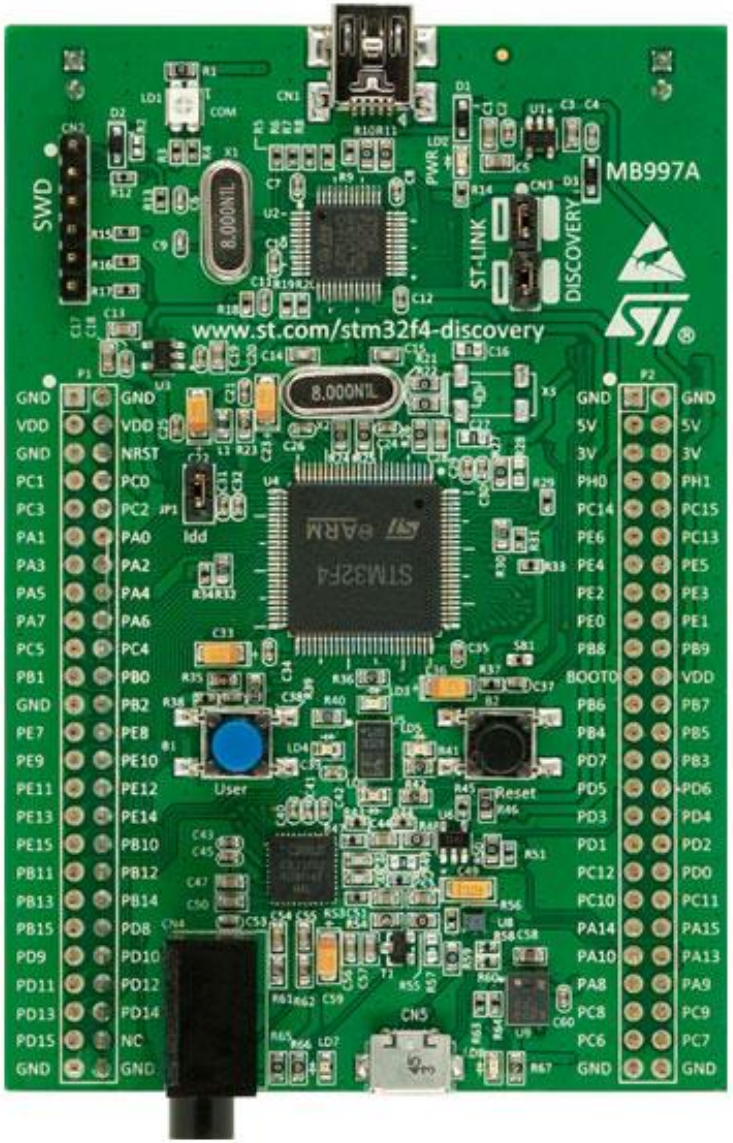
STM32F407

Features

- STM32F407VGT6 microcontroller featuring 32-bit ARM Cortex®-M4 with FPU core, 1 MB Flash memory, 192 KB RAM in an LQFP100 package
- On-board ST-LINK/V2 with selection mode switch to use the kit as a standalone STLINK/ V2 (with SWD connector for programming and debugging)
- Board power supply: through USB bus or from an external 5 V supply voltage
- External application power supply: 3 V and 5 V
- LIS302DL or LIS3DSH ST MEMS 3-axis accelerometer
- MP45DT02 ST MEMS audio sensor omni-directional digital microphone
- CS43L22 audio DAC with integrated class D speaker driver



Features

• Eight LEDs:
– LD1 (red/green) for USB communication
– LD2 (red) for 3.3 V power on
– Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)
– 2 USB OTG LEDs LD7 (green) VBus and LD8 (red) over-current

• Two push buttons (user and reset)

• USB OTG FS with micro-AB connector

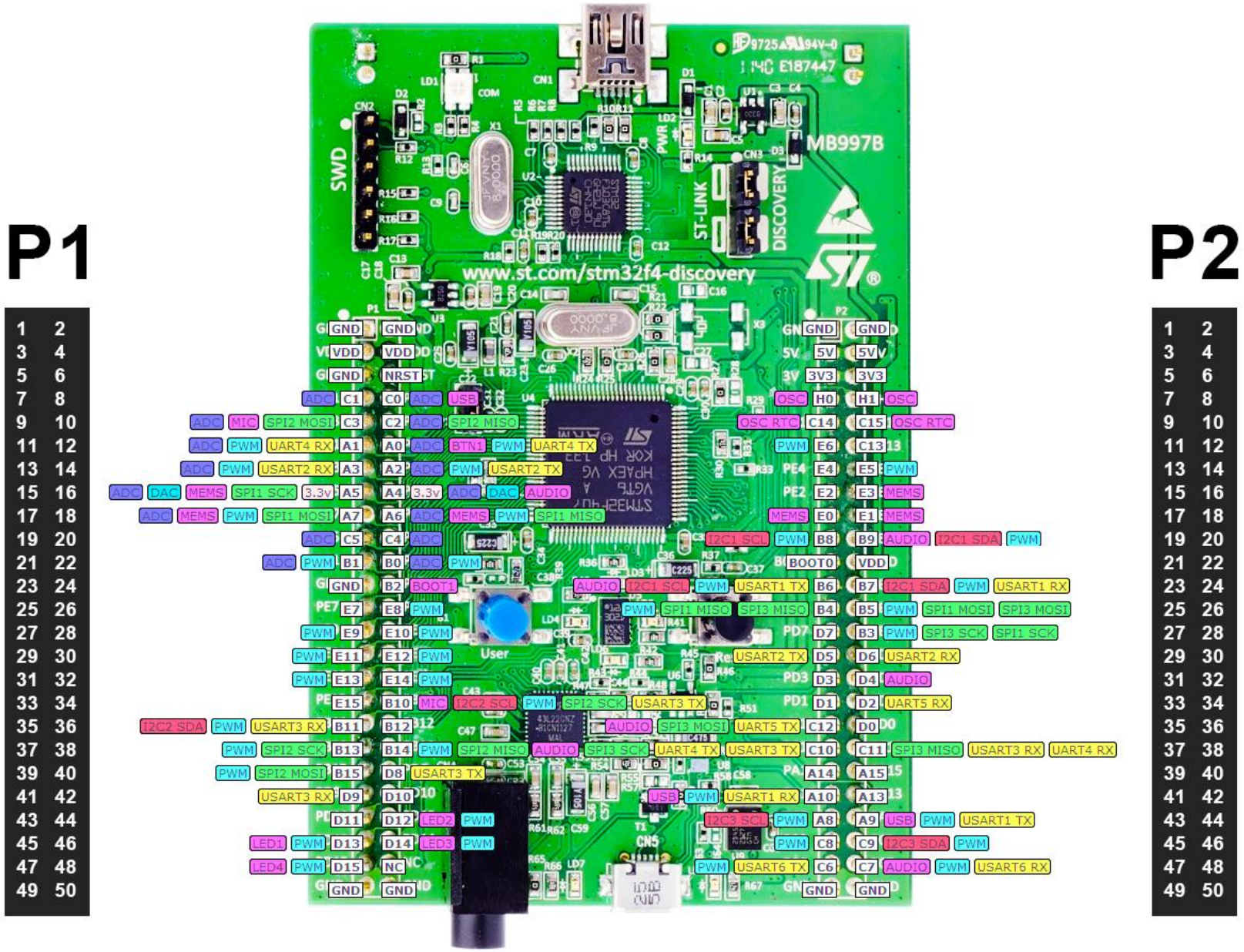
• Extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing

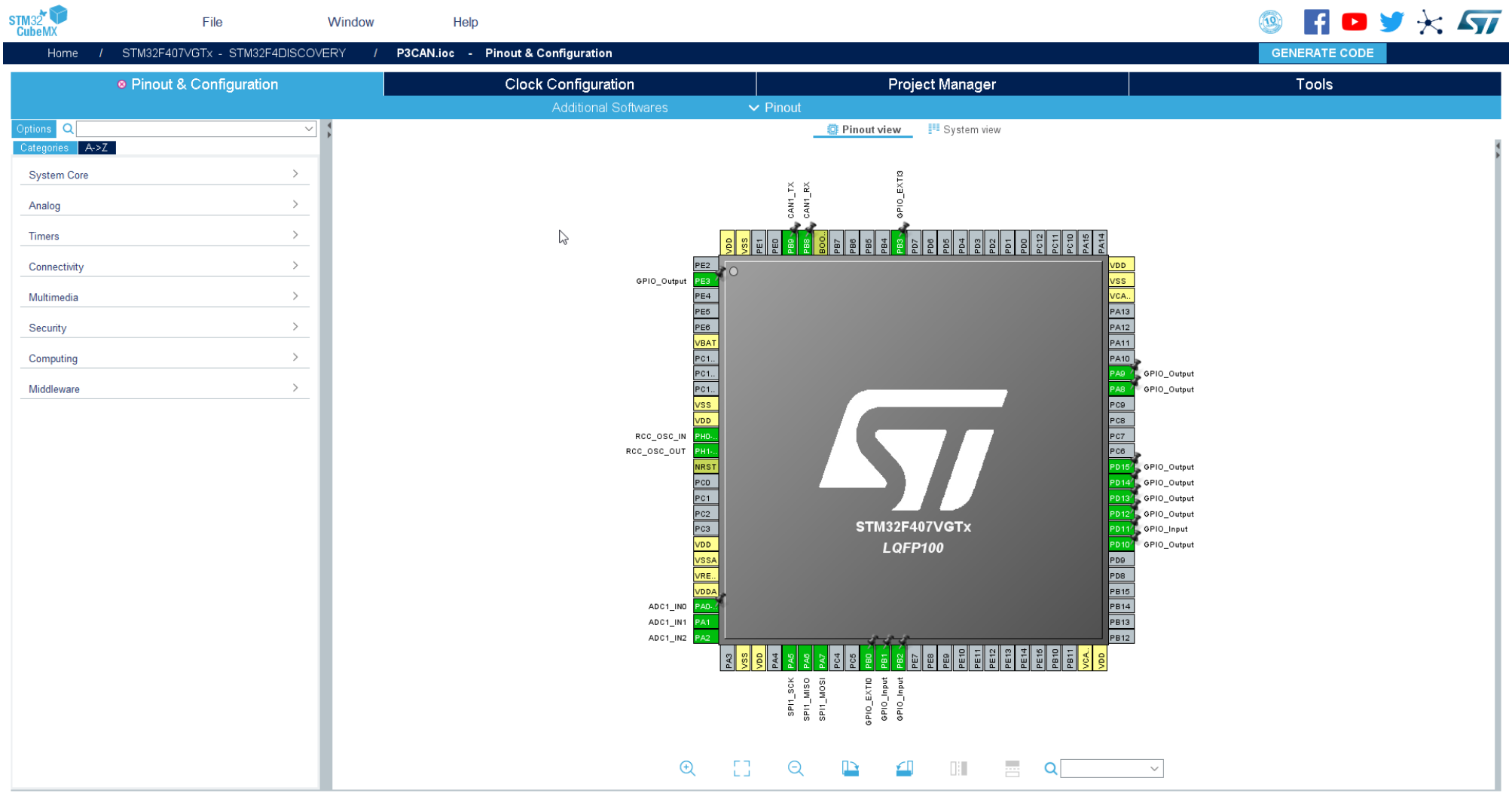
• Comprehensive free software including a variety of examples, part of STM32CubeF4 package or STSW-STM32068 for legacy standard libraries usage

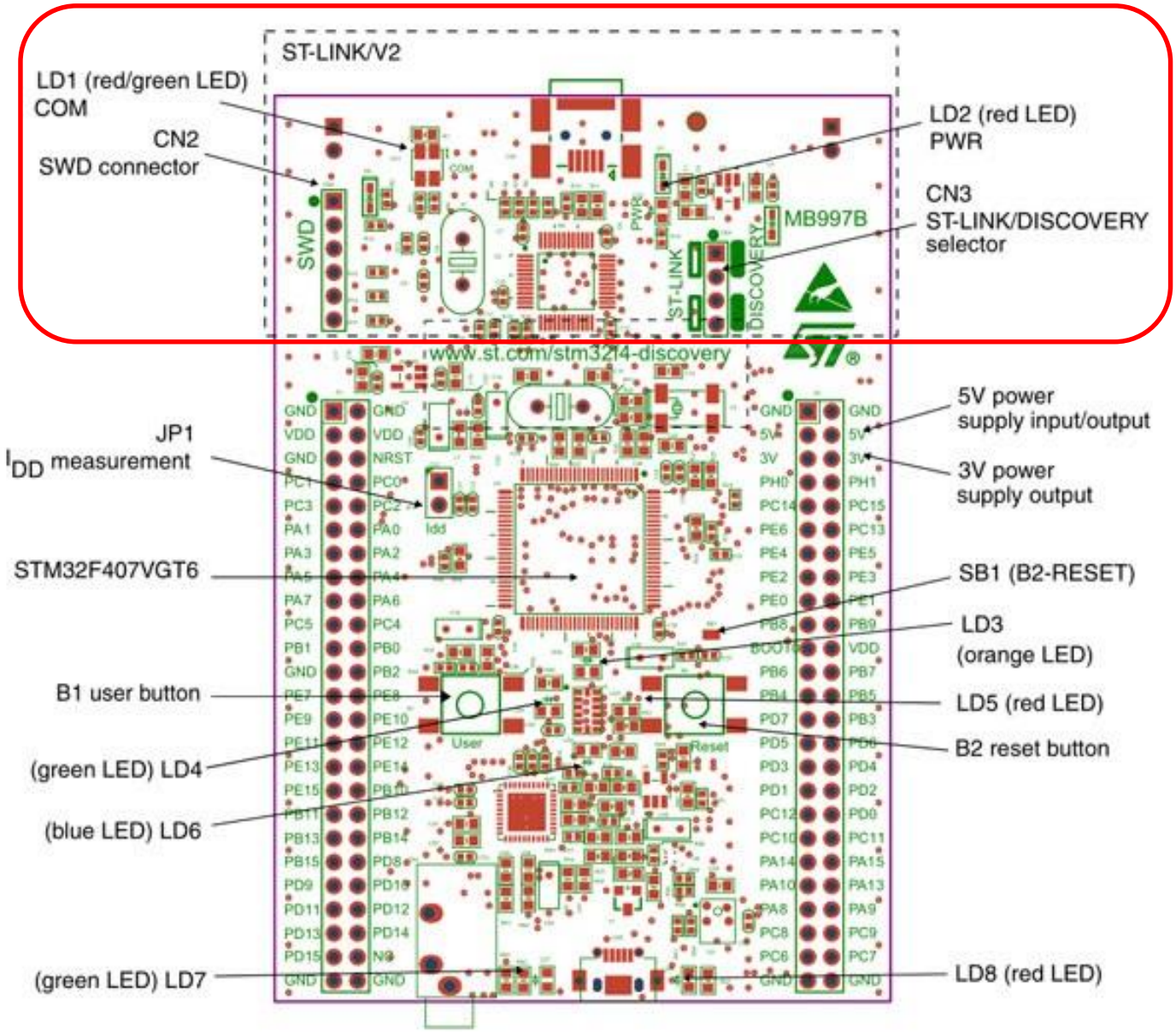
Bus SPI (Acelerómetro)

Bus I2C

Bus CAN

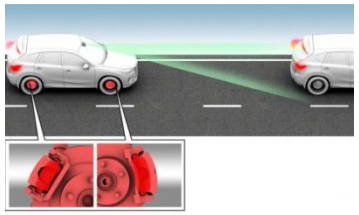

P1
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
41 42
43 44
45 46
47 48
49 50
P2
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
41 42
43 44
45 46
47 48
49 50
SWD
ST-LINK
DISCOVERY
MB997B
www.st.com/stm32f4-discovery
User
CN5


STM32 CubeMX
File
Window
Help
Home / STM32F407VGTx - STM32F4DISCOVERY / P3CAN.ioc - Pinout & Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Additional Softwares
Pinout
Options
Categories
A->Z
System Core
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware
Pinout view
System view
STM32F407VGTx
LQFP100
CAN1_TX
CAN1_RX
GPIO_EXTI3
GPIO_Output
RCC_OSC_IN
RCC_OSC_OUT
ADC1_IN0
ADC1_IN1
ADC1_IN2
SPI1_SCK
SPI1_MISO
SPI1_MOSI
GPIO_EXTI0
GPIO_Input
GPIO_Input
GPIO_Output
GPIO_Output
GPIO_Output
GPIO_Output
GPIO_Output
GPIO_Input
GPIO_Output


ST-LINK/V2
LD1 (red/green LED)
COM
CN2
SWD connector
LD2 (red LED)
PWR
CN3
ST-LINK/DISCOVERY
selector
MB997B
SWD
ST-LINK
DISCOVERY
www.st.com/stm32f4-discovery
JP1
IDD measurement
STM32F407VGT6
B1 user button
(green LED) LD4
(blue LED) LD6
(green LED) LD7
5V power
supply input/output
3V power
supply output
SB1 (B2-RESET)
LD3
(orange LED)
LD5 (red LED)
B2 reset button
LD8 (red LED)

Conexiones para Sistema Detección Distracciones

GPIO: PD12,13,14,15



GPIO: PA8, PA9
(Luz de freno)

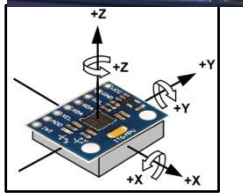
Agarre
GPIO: PB1
(GPIO PB2)

Cambio de modo
GPIO-Interr: PB0

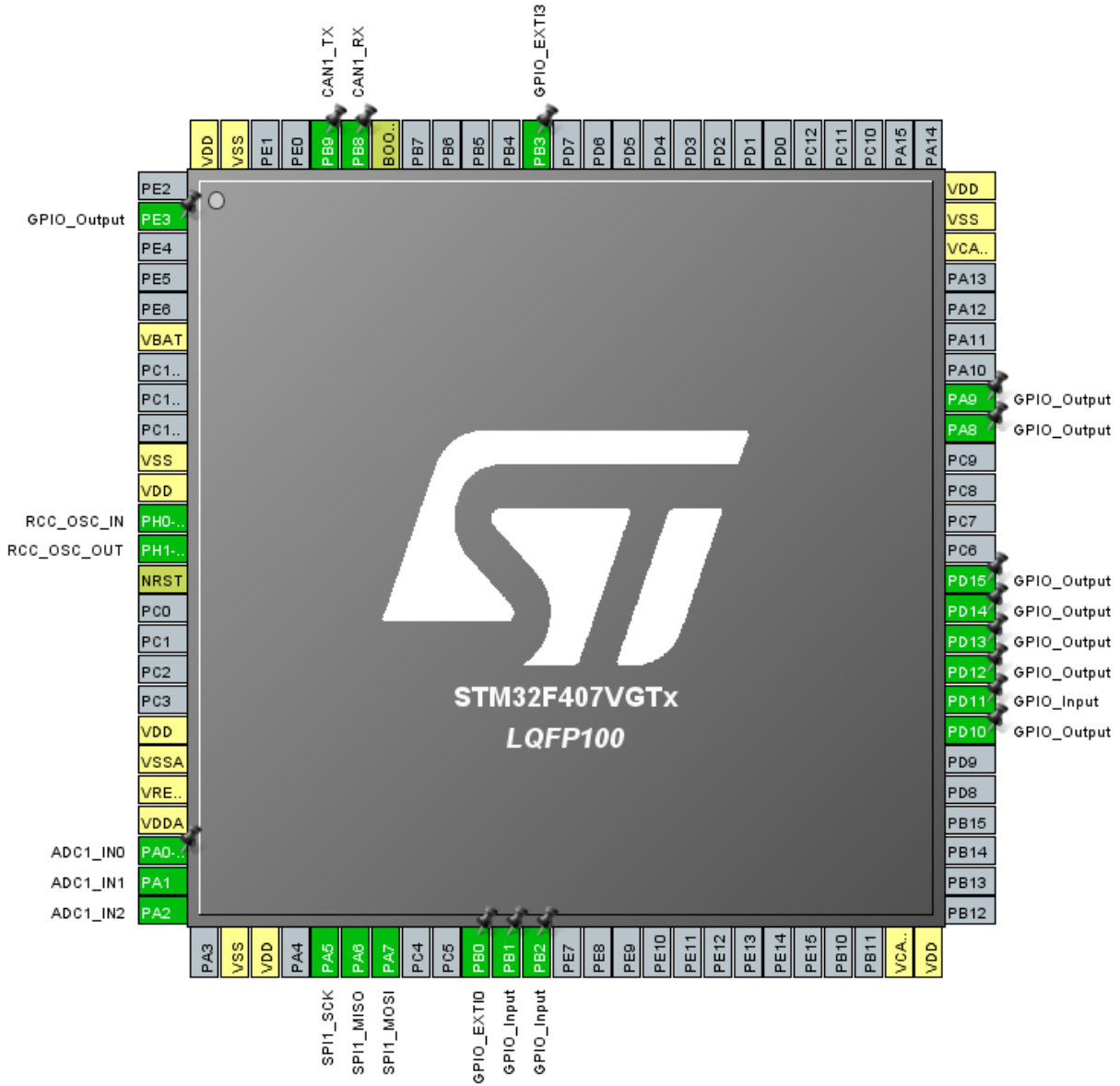
Trigger: PD10
Eco: PD11

ADC1_IN2 PA2
(ADC1_IN2 PA2)

ADC1_IN0: PA0



GPIO_Ext_IO: PE0
GPIO_OUT: PE3
SPI1_SCK: PA5
SPI1_MISO: PA6
SPI1_MOSI: PA7


CAN1_TX
CAN1_RX
GPIO_EXTI3
VDD
VSS
PE1
PE0
PB9
PB8
BOO..
PB7
PB6
PB5
PB4
PB3
PD7
PD6
PD5
PD4
PD3
PD2
PD1
PD0
PC12
PC11
PC10
PA15
PA14
PE2
GPIO_Output
PE3
PE4
PE5
PE6
VBAT
PC1..
PC1..
PC1..
VSS
VDD
RCC_OSC_IN
PH0-..
RCC_OSC_OUT
PH1-..
NRST
PC0
PC1
PC2
PC3
VDD
VSSA
VRE..
VDDA
ADC1_IN0
PA0-..
ADC1_IN1
PA1
ADC1_IN2
PA2
VDD
VSS
VCA..
PA13
PA12
PA11
PA10
PA9
GPIO_Output
PA8
GPIO_Output
PC9
PC8
PC7
PC6
PD15
GPIO_Output
PD14
GPIO_Output
PD13
GPIO_Output
PD12
GPIO_Output
PD11
GPIO_Input
PD10
GPIO_Output
PD9
PD8
PB15
PB14
PB13
PB12
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
PB10
PB11
VCA..
VDD
SPI1_SCK
SPI1_MISO
SPI1_MOSI
GPIO_EXTI0
GPIO_Input
GPIO_Input
STM32F407VGTx
LQFP100

Conexiones para Sistema Detección Distracciones

GPIO out PD12,13,14,15 – Leds luces de aviso

GPIO out PA8, PA9 – Leds para activación del freno

GPIO in PB1 – Agarre del volante
GPIO in PB2

GPIO-Interr:PB0 - Cambio de modo
GPIO-Interr:PB3

GPIO out PD10 – Trigger ultrasonidos
GPIO in PD11 – Eco ultrasonidos

ADC1_IN0 PA0 – Giros de volante
ADC1_IN1 PA1
ADC1_IN2 PA2 - Velocímetro

GPIO_Ext_IO PE0 - Acelerómetro
GPIO_OUT PE3
SPI1_SCK PA5
SPI1_MISO PA6
SPI1_MOSI PA7

```
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
//Encender led naranja D13

HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
//Apagar led rojo D14

HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
//Cambiar estado del led D15

if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_1) == GPIO_PIN_SET)
// Leer el valor del pin B1
```

CÓDIGO PARA LA LECTURA DE UN CANAL ANALÓGICO DENTRO DE UNA TAREA

```
int actual = 0; //Valor actual

/* Lectura del canal ADC0 */
ADC_ChannelConfTypeDef sConfig = {0};

sConfig.Channel = ADC_CHANNEL_0; // seleccionamos el canal 0
sConfig.Rank = 1;
sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES;

HAL_ADC_ConfigChannel(&hadc1, &sConfig);
HAL_ADC_Start(&hadc1); // comenzamos la conversón AD

if(HAL_ADC_PollForConversion(&hadc1, 5) == HAL_OK){
        actual = HAL_ADC_GetValue(&hadc1); // leemos el valor
        PosicionVolante = actual; // actualizamos una variable global
}
```

```
void StartUltrasoundTask(void const * argument)
{
  TickType_t xLastWakeTime;

for(;;)
  {
    xLastWakeTime = xTaskGetTickCount();

    currentDistance = (float)readDistance() * 0.00171821;
    if (currentDistance == 500000) currentDistance = 1;

    vTaskDelayUntil( &xLastWakeTime, pdMS_TO_TICKS( 400 )); //Periodo de 400mS
  }
}
```

```
uint32_t readDistance(void)
{
        __IO uint8_t flag=0;
        __IO uint32_t disTime=0;

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_SET);
        DWT_Delay_us(10);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_RESET);

        while(flag == 0){
                while(HAL_GPIO_ReadPin(GPIOD,GPIO_PIN_11) == GPIO_PIN_SET){
                        disTime++;
                        flag = 1;
                }
        }
        return disTime;
}
```

```
uint32_t DWT_Delay_Init(void) {
  /* Disable TRC */
  CoreDebug->DEMCR &= ~CoreDebug_DEMCR_TRCENA_Msk; // ~0x01000000;
  /* Enable TRC */
  CoreDebug->DEMCR |=  CoreDebug_DEMCR_TRCENA_Msk; // 0x01000000;

  /* Disable clock cycle counter */
  DWT->CTRL &= ~DWT_CTRL_CYCCNTENA_Msk; //~0x00000001;
  /* Enable  clock cycle counter */
  DWT->CTRL |=  DWT_CTRL_CYCCNTENA_Msk; //0x00000001;

  /* Reset the clock cycle counter value */
  DWT->CYCCNT = 0;

     /* 3 NO OPERATION instructions */
     __ASM volatile ("NOP");
     __ASM volatile ("NOP");
  __ASM volatile ("NOP");

  /* Check if clock cycle counter has started */
     if(DWT->CYCCNT)
     {
       return 0; /*clock cycle counter started*/
     }
     else
  {
    return 1; /*clock cycle counter not started*/
  }
}
```