ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

8. Analog Joystick

Lead Partner: Warsaw University of Technology

Authors: Daniel Krol

University of Applied Sciences in Tarnow

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

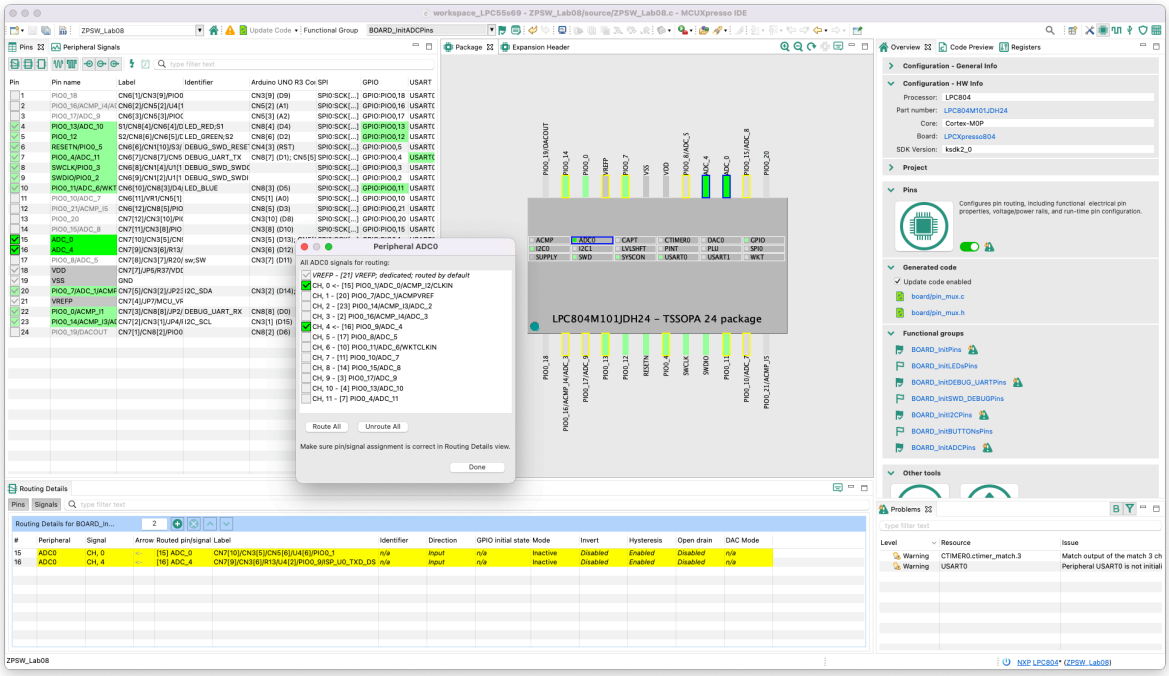
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

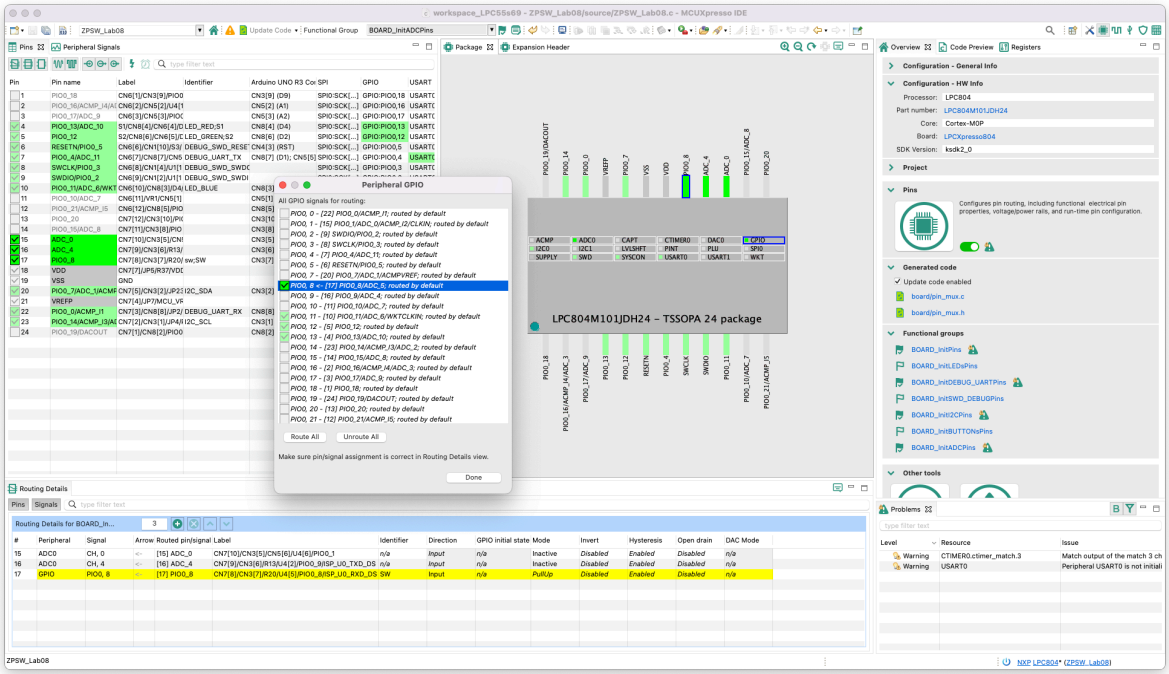
8. Analog Joystick

I. A/D Converter

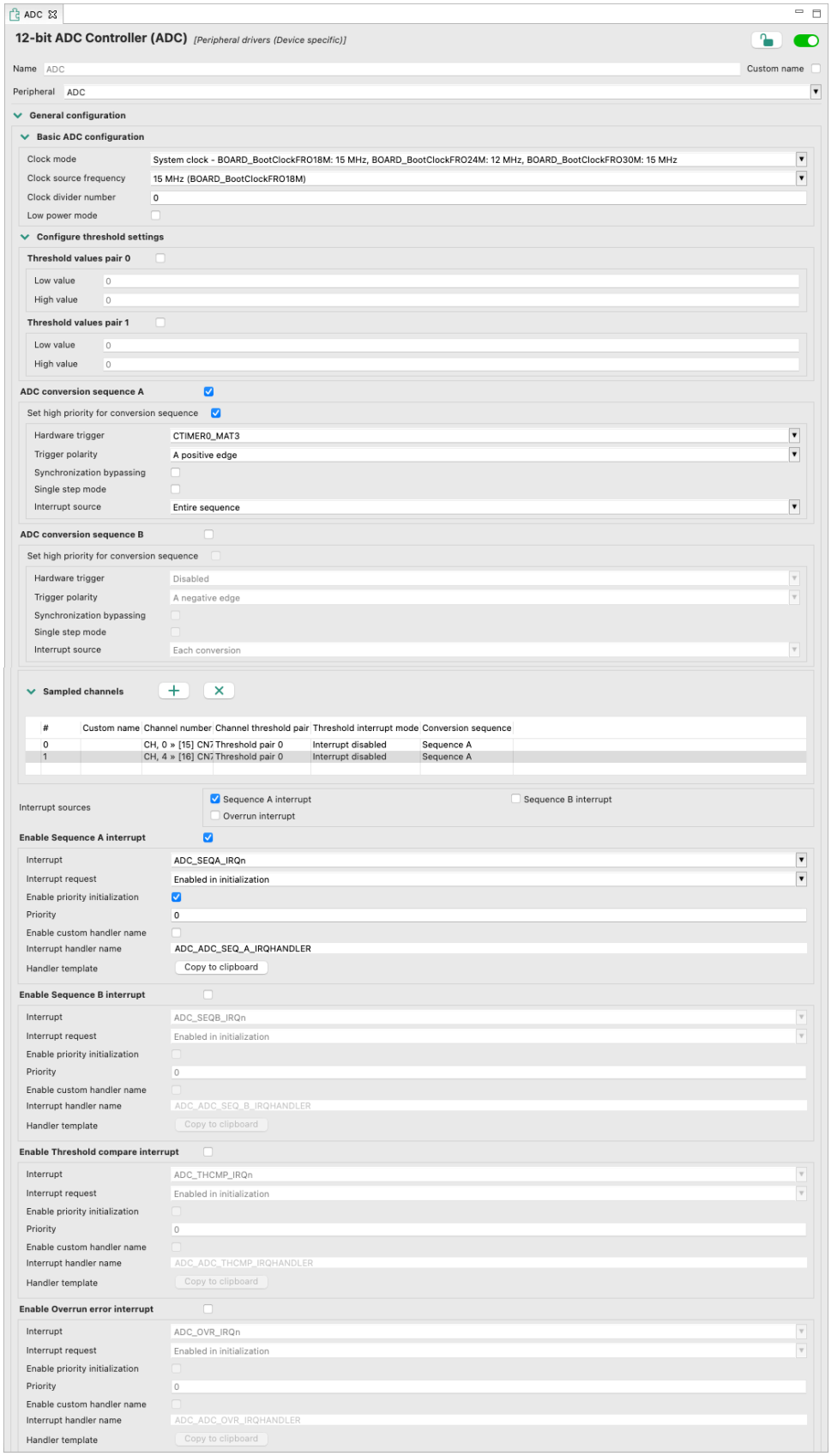
1. Copy the project from the previous class and name it eg *Lab08*.
2. Go to *Config Tool -> Pins* and open the *BOARD_InitADCPins* preset. Click on the ADC block and to the existing *ADC0* signal (*PIO0_1* pinout), similarly add the *ADC4* signal (*PIO0_9* pinout):



3. Add pin *PIO0_8* as input from *PullUp* and add *SW* identifier:



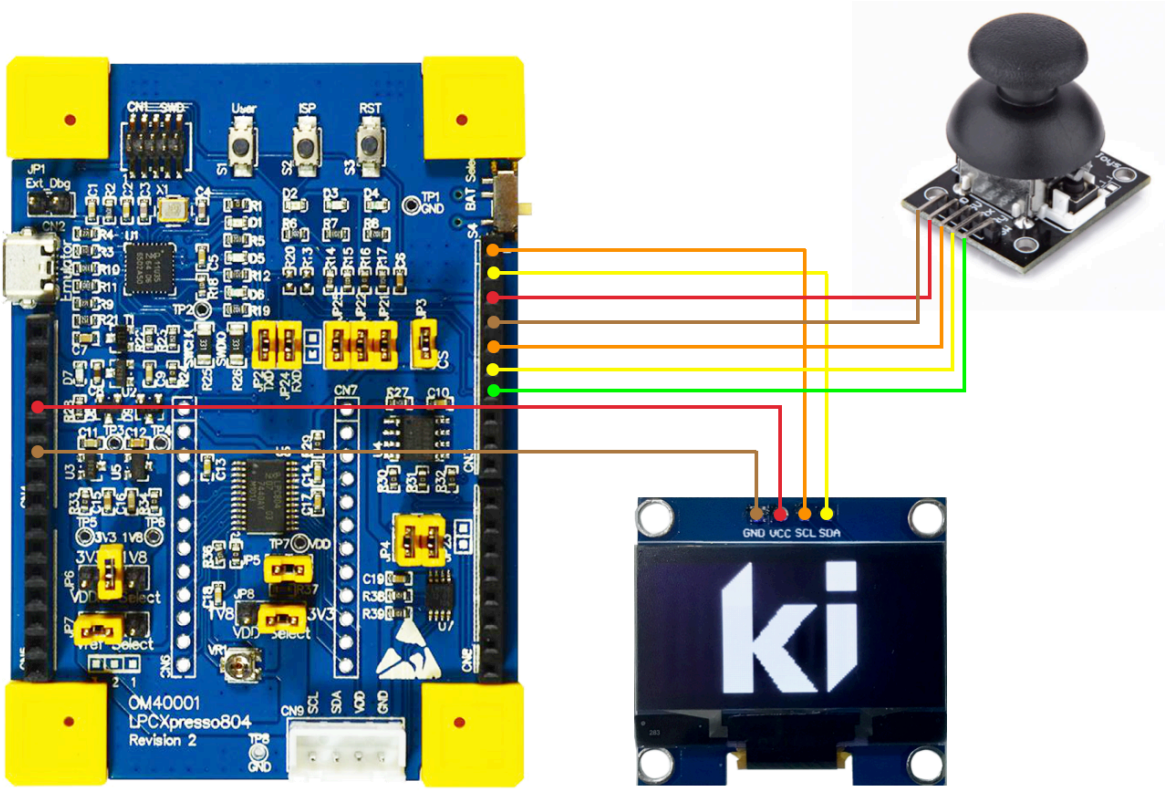
4. Go to the ADC settings and change its configuration by adding an additional channel (CH 4):



5. Go to the main project file and modify the code as below:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "fsl_power.h"
#include "oled.h"

static adc_result_info_t gAdcResultInfoStruct;
adc_result_info_t *volatile gAdcResultInfoPtr = &gAdcResultInfoStruct;

char sbuff[32];

volatile uint16_t gAxisX = 0;
volatile uint16_t gAxisY = 0;


/* ADC_SEQA_IRQn interrupt handler */
void ADC_ADC_SEQ_A_IRQHANDLER(void) {
          /* Get status flags */
          if (kADC_ConvSeqAInterruptFlag == (kADC_ConvSeqAInterruptFlag & ADC_GetStatusFlags(ADC_PERIPHERAL))) {
                    /* Place your interrupt code here */
                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 0, gAdcResultInfoPtr);
                    gAxisY = gAdcResultInfoStruct.result;

                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                    gAxisX = gAdcResultInfoStruct.result;

                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
          }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Power on ADC. */
          POWER_DisablePD(kPDRUNCFG_PD_ADC0);
          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
          /* Initialize OLED */
          OLED_Init(I2C0_PERIPHERAL);

          while(1) {

                    OLED_Clear_Screen(0);
                    sprintf(sbuff, "X: %5d", gAxisX);
                    OLED_Puts(0, 0, sbuff);
                    sprintf(sbuff, "Y: %5d", gAxisY);
                    OLED_Puts(0, 1, sbuff);
                    OLED_Refresh_Gram();
          }
          return 0 ;
}
```

6. Connect the display and the joystick to the board according to the following diagram:



7. Program the microcontroller and check the example operation.

II. Button operation

1. Modify your project code by adding Z axis button support:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "fsl_power.h"
#include "oled.h"

static adc_result_info_t gAdcResultInfoStruct;
adc_result_info_t *volatile gAdcResultInfoPtr = &gAdcResultInfoStruct;
char sbuff[32];
volatile uint16_t gAxisX = 0;
volatile uint16_t gAxisY = 0;
volatile bool     gAxisZ = 0;


/* ADC_SEQA_IRQn interrupt handler */
void ADC_ADC_SEQ_A_IRQHANDLER(void) {
          /* Get status flags */
          if (kADC_ConvSeqAInterruptFlag == (kADC_ConvSeqAInterruptFlag & ADC_GetStatusFlags(ADC_PERIPHERAL))) {
                    /* Place your interrupt code here */
                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 0, gAdcResultInfoPtr);
                    gAxisY = gAdcResultInfoStruct.result;

                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                    gAxisX = gAdcResultInfoStruct.result;

                    gAxisZ = GPIO_PinRead(BOARD_INITADCPINS_SW_GPIO,
                                          BOARD_INITADCPINS_SW_PORT,
                                          BOARD_INITADCPINS_SW_PIN);

                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
```

```
        }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

        /* Power on ADC. */
        POWER_DisablePD(kPDRUNCFG_PD_ADC0);
        /* Init board hardware. */
        BOARD_InitBootPins();
        BOARD_InitBootClocks();
        BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
        /* Init FSL debug console. */
        BOARD_InitDebugConsole();
#endif
        /* Initialize OLED */
        OLED_Init(I2C0_PERIPHERAL);

        while(1) {

                OLED_Clear_Screen(0);
                sprintf(sbuff, "X: %5d", gAxisX);
                OLED_Puts(0, 0, sbuff);
                sprintf(sbuff, "Y: %5d", gAxisY);
                OLED_Puts(0, 1, sbuff);
                sprintf(sbuff, "Z: %5d", gAxisZ);
                OLED_Puts(0, 2, sbuff);
                OLED_Refresh_Gram();
        }
        return 0 ;
}
```

2. Build the project in ***Release*** mode, program the microcontroller and check the example.

III. Cursor support

1. Modify the project code:

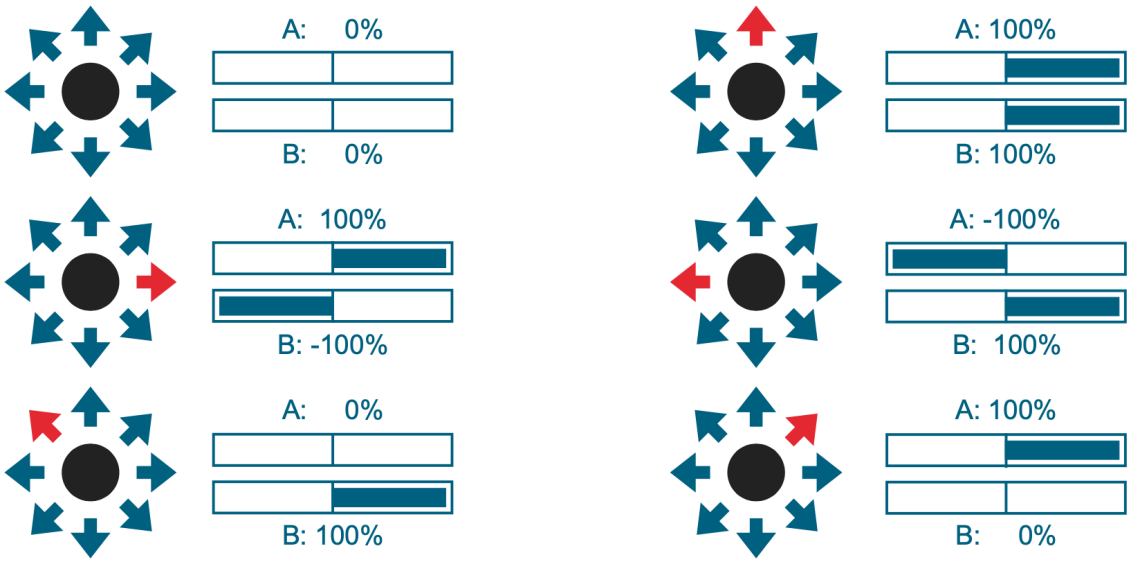
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "fsl_power.h"
#include "oled.h"


static adc_result_info_t gAdcResultInfoStruct;
adc_result_info_t *volatile gAdcResultInfoPtr = &gAdcResultInfoStruct;
char sbuff[32];
volatile uint16_t gAxisX = 0;
volatile uint16_t gAxisY = 0;
volatile bool     gAxisZ = 0;


/* ADC_SEQA_IRQn interrupt handler */
void ADC_ADC_SEQ_A_IRQHANDLER(void) {
        /* Get status flags */
        if (kADC_ConvSeqAInterruptFlag == (kADC_ConvSeqAInterruptFlag & ADC_GetStatusFlags(ADC_PERIPHERAL))) {
                /* Place your interrupt code here */
                ADC_GetChannelConversionResult(ADC_PERIPHERAL, 0, gAdcResultInfoPtr);
                gAxisY = gAdcResultInfoStruct.result;

                ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                gAxisX = gAdcResultInfoStruct.result;

                gAxisZ = GPIO_PinRead(BOARD_INITADCPINS_SW_GPIO,
                                      BOARD_INITADCPINS_SW_PORT,
                                      BOARD_INITADCPINS_SW_PIN);

                /* Clear status flags */
                ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
        }
}

void setCursor(uint8_t x, uint8_t y, uint8_t size) {

        int8_t a, b;

        a=x-size;
        b=x+size;
        if(a<0) {
                a=0;
        }
        OLED_Draw_Line(a, y, b, y);
        a=y-size;
```

```
		b=y+size;
		if(a<0) {
			a=0;
		}
		OLED_Draw_Line(x, a, x, b);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

		uint8_t cx, cy;

		/* Power on ADC. */
		POWER_DisablePD(kPDRUNCFG_PD_ADC0);
		/* Init board hardware. */
		BOARD_InitBootPins();
		BOARD_InitBootClocks();
		BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
		/* Init FSL debug console. */
		BOARD_InitDebugConsole();
#endif
		/* Initialize OLED */
		OLED_Init(I2C0_PERIPHERAL);

		while(1) {

			cx = gAxisX/32;    // width: 128
			cy = 63-gAxisY/64; // height: 64

			OLED_Clear_Screen(0);
			sprintf(sbuff, "X:%3d Y:%2d Z:%d", cx, cy, gAxisZ);
			OLED_Puts(0, 0, sbuff);

			setCursor(cx, cy, 5);
			if(!gAxisZ) {
				OLED_Draw_Circle(cx, cy, 8);
			}
			OLED_Refresh_Gram();
		}
		return 0 ;
}
```

2. Build the project in ***Release*** mode, program the microcontroller and check the example.

IV. Exercises

1. Write a *PowerControl* function capable of generating control signals for 2 motors of the tracked vehicle depending on the position of the joystick. The function should present the calculated control in the form of two progress bars or deflection indicators (as in the previous class) and display the power values as a percentage. Examples of joystick settings:



Programing of embedded systems

8. Analog Joystick

In order to display negative values, integer variables with printf functions, sprint etc., you should add the *PRINTF_ADVANCED_ENABLE* constant in the preprocessor settings: