Teaching Microcontrollers with MSP430

Humberto Santos, Pedro Dinis,

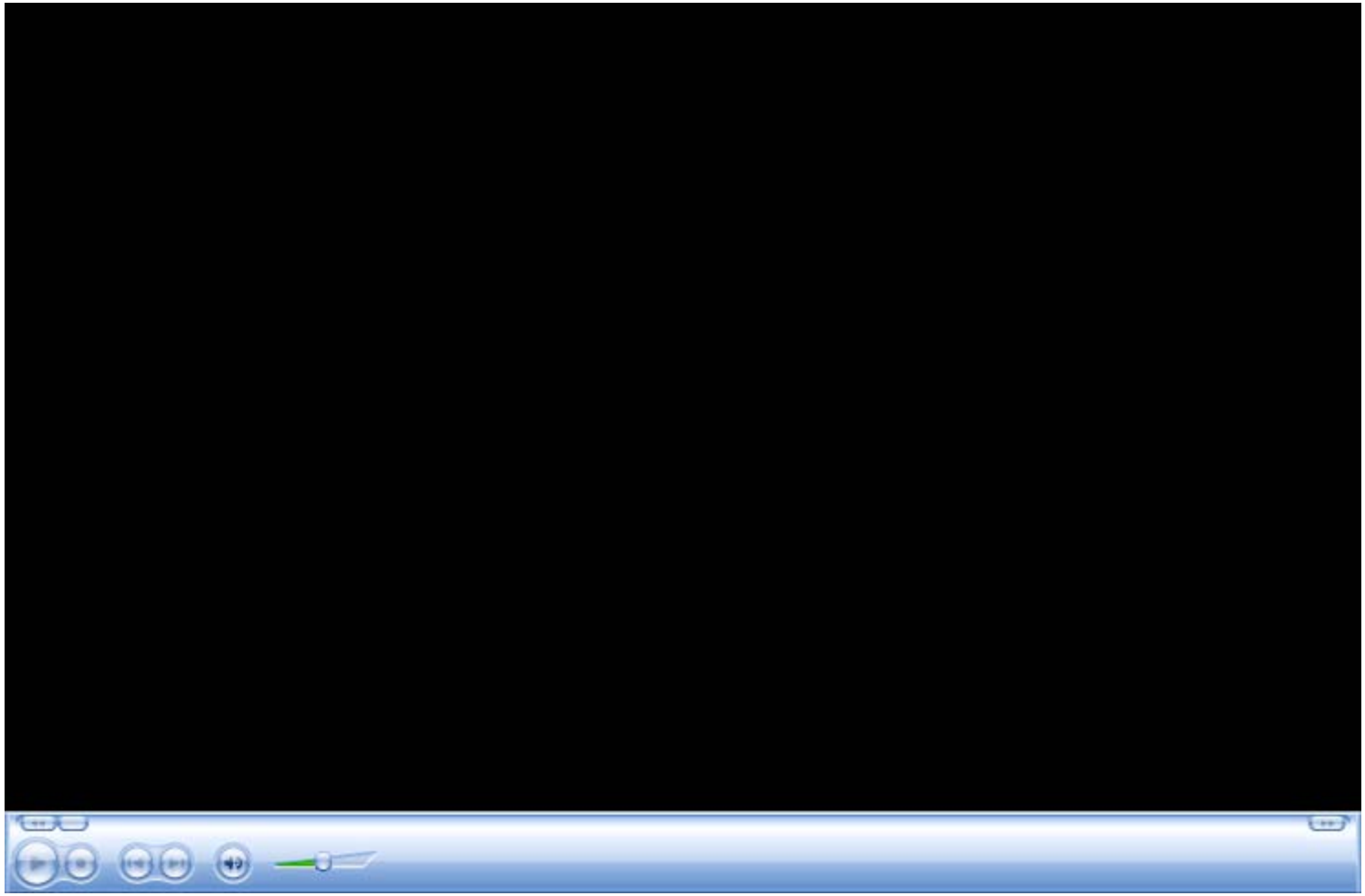
António Espírito Santo, Bruno Ribeiro

Engineer/Assistant Professor

Agenda

- Why MSP430 in classes
- Choosing development tools
- Microprocessor teaching at UBI
- Microprocessor introduction
- Examples of microprocessor usage in other classes
- Advanced microprocessor applications

Where we come from?



Why MSP430?

- cost;
- plenty of free development tools;
- good hardware start up kits and inexpensive;
- diverse peripherals in the microprocessor giving the students many abilities in the area;
- rapid learning curve;
- low power features;
- high pinout counting;
- C and C++ easy programming.

Choosing Hardware

- There are various companies that develop good development kits, some of then are:

Texas Instruments OLIMEX SoftBaugh™ IAR SYSTEMS

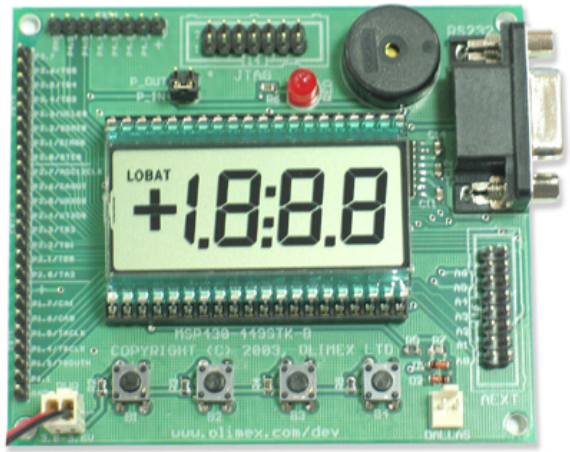
We Choose: OLIMEX

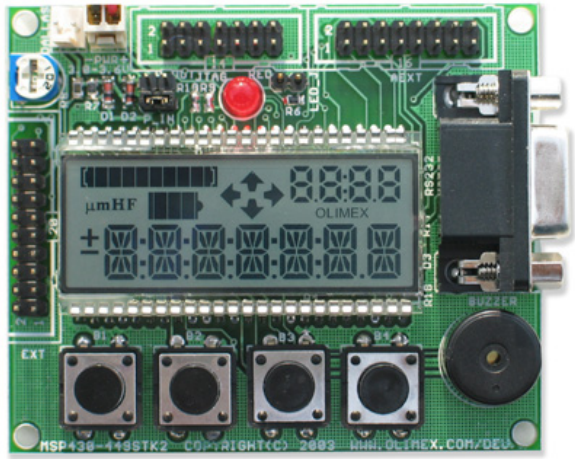
OLIMEX Advantages:

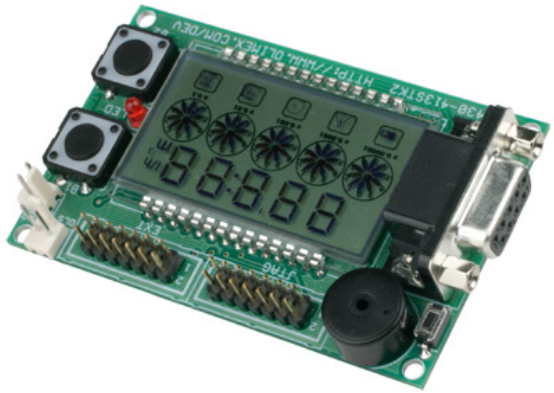
- price;
- many peripherals in the kits.

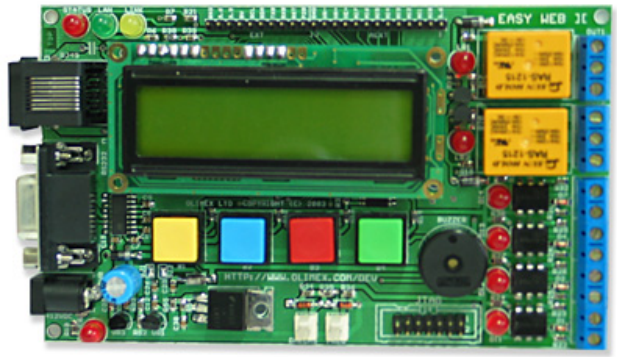
OLIMEX kits

Some OLIMEX kits used in the classes:









Choosing Software

- Some software that compile MSP430 C/C++ code:

Free
Limited to 8 KB

IAR SYSTEMS

Free
Limited to 4 KB

CrossWorks for MSP430

Not Free
30 day demo available

Free
unlimited

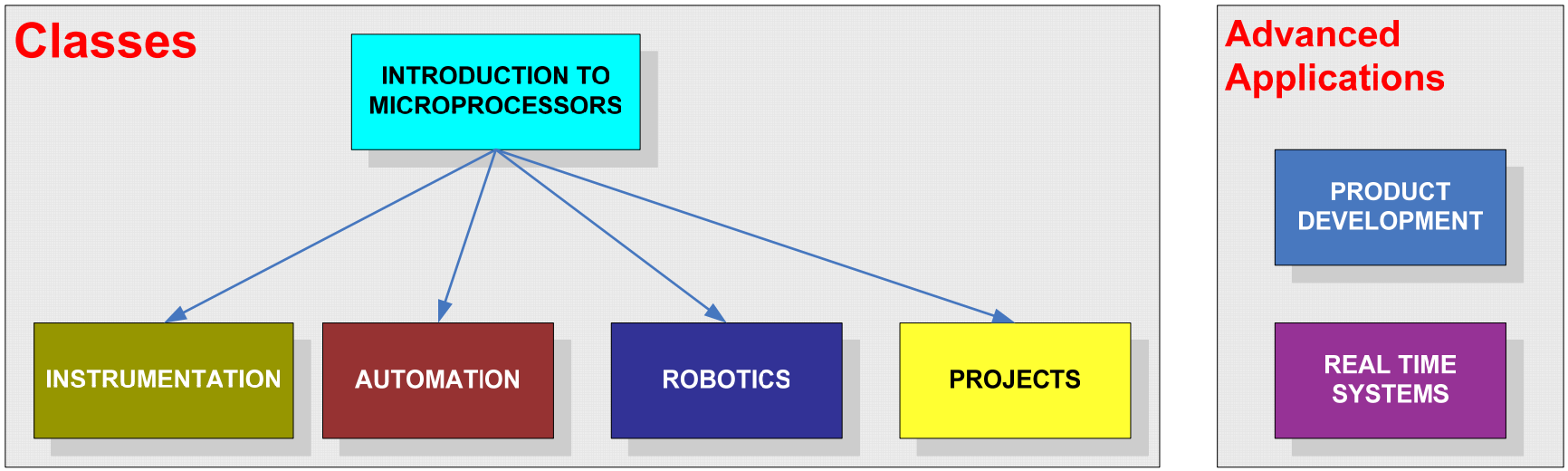
We choose:

- IAR Kickstart Version for the Classes but shifting right now to Code Composer Essentials;
- MSPGCC for R&D and projects that require more than 4KB code.

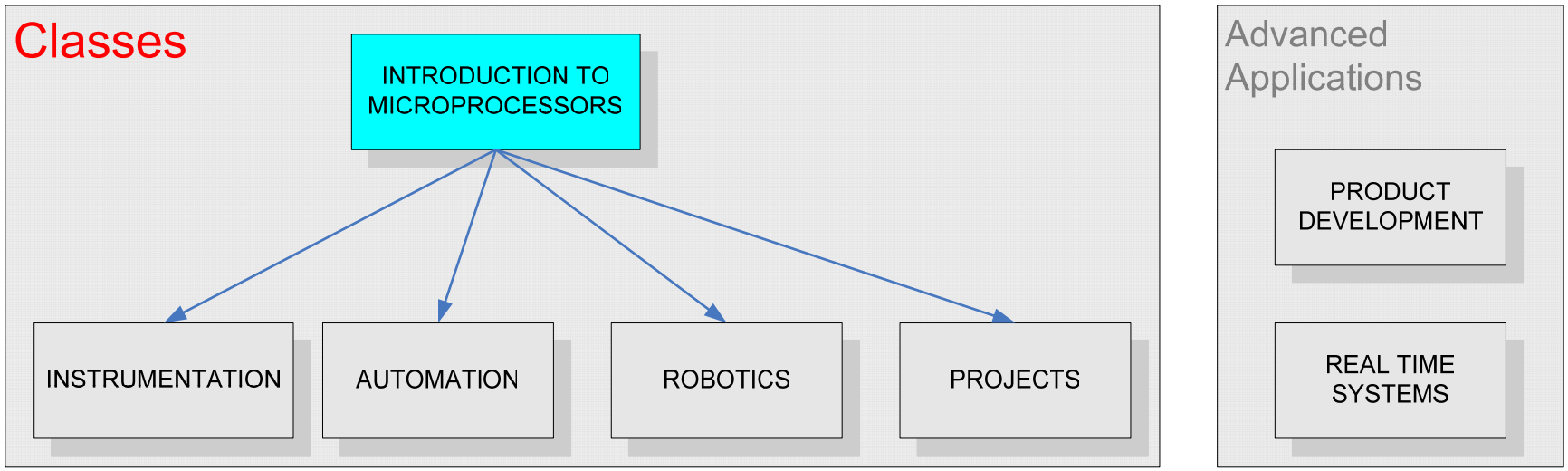
Students pre-requisites

- digital/analog electronic background;
- C/C++ programming background;
- assembler programming background;
- generic microprocessor architecture;
- software architecture for real time systems.

MSP430@UBI

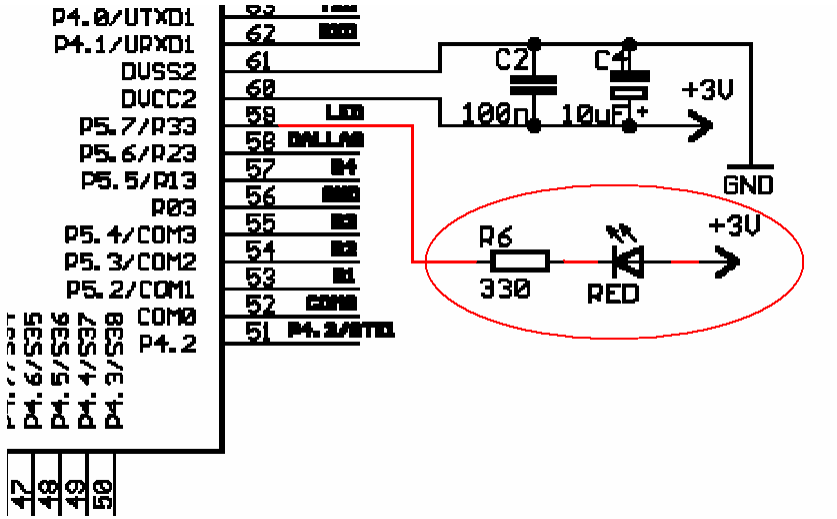

Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

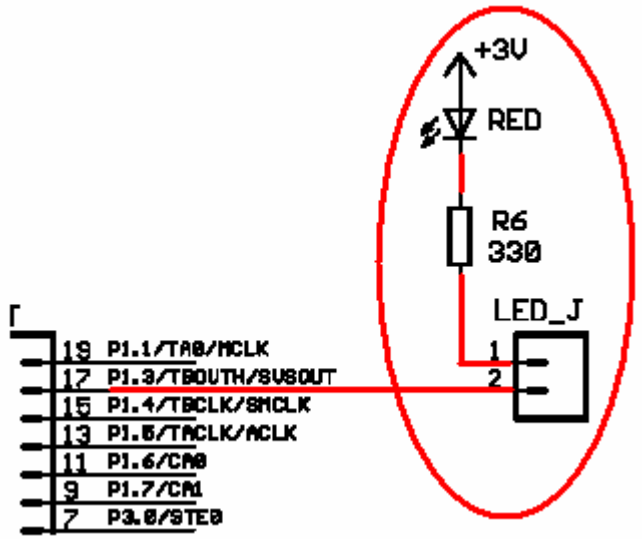
Introduction to Microprocessors



Power the LED on board (1)

- analysis of the OLIMEX kit schema with MSP430F449 (STK and STK2), to identify at which port is connected the LED;





Datasheet:

kit OLIMEX MSP430F449 STK

kit OLIMEX MSP430F449 STK2

Power the LED on board (2)

- search in the TI MSP430 Manual for the chapter related to ports configuration:
  - explanation about the ports characteristics: registers;
  - identify the registers needed to configure the ports for this application;
  - explanation with both hexadecimal and definition file mnemonics configuration;

Chapter 9

Digital I/O

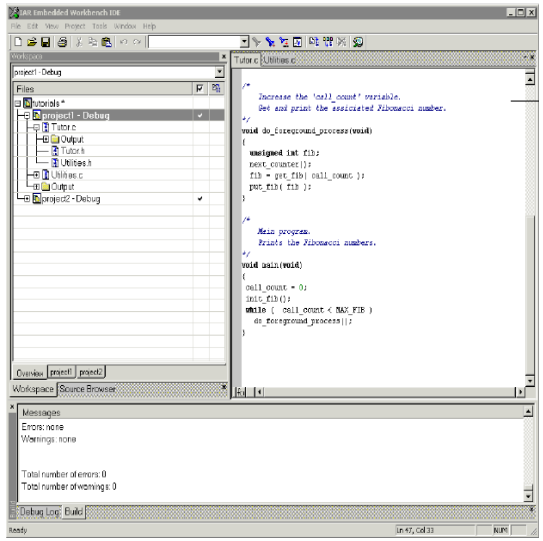
This chapter describes the operation of the digital I/O ports. Ports P1-P6 are implemented in all MSP430x4xx devices.

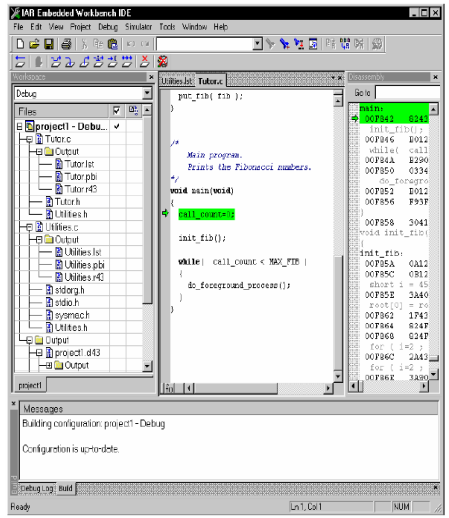
Topic		Page
9.1	Digital I/O Introduction	9-2
9.2	Digital I/O Operation	9-3
9.3	Digital I/O Registers	9-7

Port	Register	Short Form	Address	Register Type	Initial State
P5	Input	P5IN	030h	Read only	–
	Output	P5OUT	031h	Read/write	Unchanged
	Direction	P5DIR	032h	Read/write	Reset with PUC
	Port Select	P5SEL	033h	Read/write	Reset with PUC

Power the LED on board (3)

- search in the IAR Systems Manual how to create and compile the file for the task:
  - introduction to IAR Embedded WorkBench;
  - explanation how to create a project in IAR Embedded WorkBench;
  - C programming specifications (particularities from C standard) to create the file;





- hands-on Lab.

LCD (1)

- timers description: Watchdog Timer and Basic Timer;
- search in the TI MSP430 Manual for the chapter related with Watchdog Timer and Basic Timer characteristics:
  - presentation of timers configuration: registers;

Chapter 10

Watchdog Timer, Watchdog Timer+

The watchdog timer is a 16-bit timer that can be used as a watchdog or as an interval timer. This chapter describes the watchdog timer. The watchdog timer is implemented in all MSP430x4xx devices. The MSP430x42x and MSP430FE42x devices implement an enhanced WDT called WDT+.

Topic Page

Table 10–1. Watchdog Timer Registers

Register	Short Form	Register Type	Address	Initial State
Watchdog timer control register	WDTCTL	Read/write	0120h	06900h with PUC
SFR interrupt enable register 1	IE1	Read/write	0000h	Reset with PUC
SFR interrupt flag register 1	IFG1	Read/write	0002h	Reset with PUC[1]

1) WDTIFG is reset with POR

Chapter 11

Basic Timer1

The Basic Timer1 module is two independent, cascadable 8-bit timers. This chapter describes the Basic Timer1. Basic Timer1 is implemented in all MSP430x4xx devices.

Topic Page

11.3 Basic Timer1 Registers

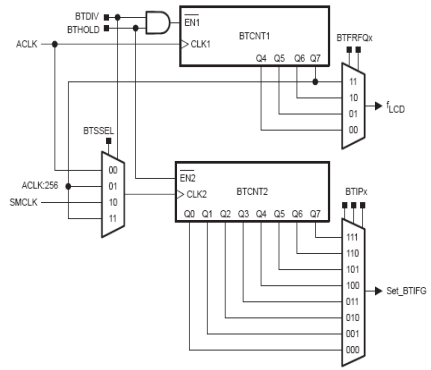
The watchdog timer module registers are listed in Table 11–1.

Table 11–1. Basic Timer1 Registers

Register	Short Form	Register Type	Address	Initial State
Basic Timer1 Control	BTCTL	Read/write	040h	Unchanged
Basic Timer1 Counter 1	BTCNT1	Read/write	046h	Unchanged
Basic Timer1 Counter 2	BTCNT2	Read/write	047h	Unchanged
SFR interrupt flag register 2	IFG2	Read/write	001h	Reset with PUC
SFR interrupt enable register 2	IE2	Read/write	003h	Reset with PUC

Note: The Basic Timer1 registers should be configured at power-up. There is no initial state for BTCTL, BTCNT1, or BTCNT2

Figure 11–1. Basic Timer1 Block Diagram



BTCTL, Basic Timer1 Control Register

7	6	5	4	3	2	1	0
BTSSEL	BTHOLD	BTDIV	BTFRFQx		BTIPx		
rw	rw	rw	rw	rw	rw	rw	rw

LCD (2)

- search in the TI MSP430 Manual for the chapter related with LCD characteristics:
  - presentation of LCD controller characteristics;
  - presentation of LCD controller configuration: control and memory registers;

Chapter 18

LCD Controller

The LCD controller drives static, 2-mux, 3-mux, or 4-mux LCDs. This chapter describes LCD controller. The LCD controller is implemented on all MSP430x4xx devices.

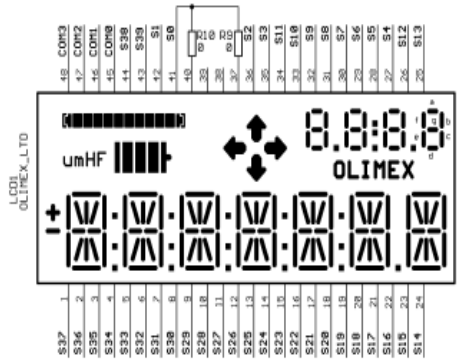
Topic Page

18.3 LCD Controller Registers

The LCD Controller registers are listed in Table 18–1.

Table 18–1.LCD Controller Registers

Register	Short Form	Register Type	Address	Initial State
LCD control register	LCDCTL	Read/write	090h	Reset with PUC
LCD memory 1	LCDM1	Read/write	091h	Unchanged
LCD memory 2	LCDM2	Read/write	092h	Unchanged
LCD memory 3	LCDM3	Read/write	093h	Unchanged
LCD memory 4	LCDM4	Read/write	094h	Unchanged
LCD memory 5	LCDM5	Read/write	095h	Unchanged
LCD memory 6	LCDM6	Read/write	096h	Unchanged
LCD memory 7	LCDM7	Read/write	097h	Unchanged
LCD memory 8	LCDM8	Read/write	098h	Unchanged
LCD memory 9	LCDM9	Read/write	099h	Unchanged
LCD memory 10	LCDM10	Read/write	09Ah	Unchanged
LCD memory 11	LCDM11	Read/write	09Bh	Unchanged
LCD memory 12	LCDM12	Read/write	09Ch	Unchanged
LCD memory 13	LCDM13	Read/write	09Dh	Unchanged
LCD memory 14	LCDM14	Read/write	09Eh	Unchanged
LCD memory 15	LCDM15	Read/write	09Fh	Unchanged
LCD memory 16	LCDM16	Read/write	0A0h	Unchanged
LCD memory 17	LCDM17	Read/write	0A1h	Unchanged
LCD memory 18	LCDM18	Read/write	0A2h	Unchanged
LCD memory 19	LCDM19	Read/write	0A3h	Unchanged
LCD memory 20	LCDM20	Read/write	0A4h	Unchanged



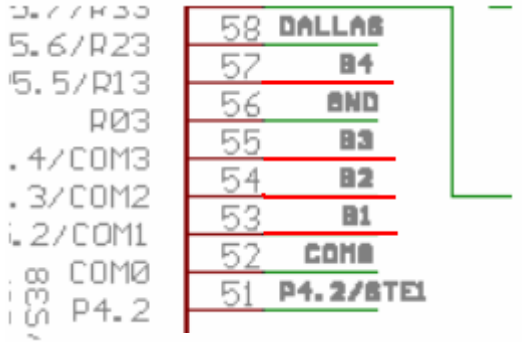
LCDCTL, LCD Control Register

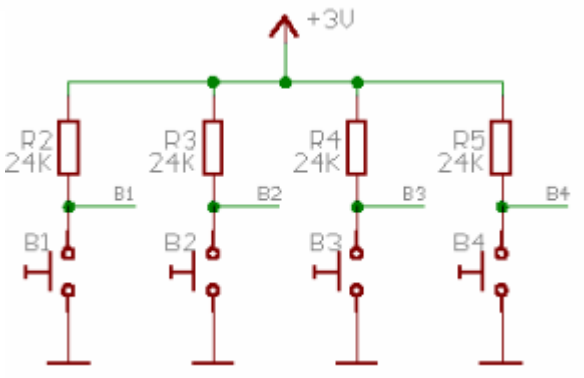
7	6	5	4	3	2	1	0
LCDPx			LCDMXx		LCDSON	Unused	LCDON
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

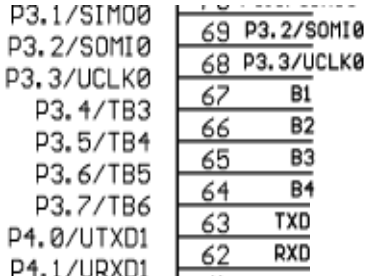
- hands-on Lab.

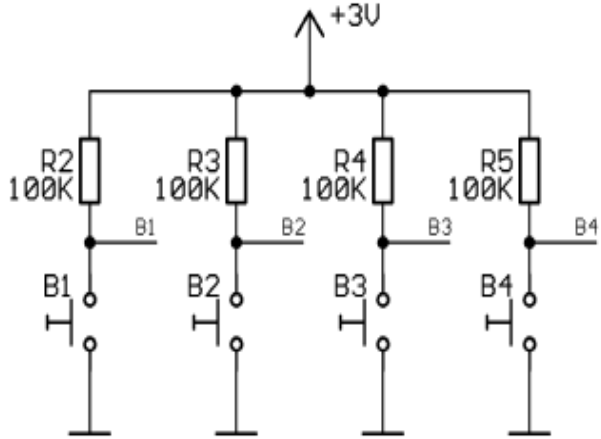
LED + buttons + LCD (1)

- analysis of OLIMEX kit schema with MSP430F449 (STK and STK2), to identify at which ports are connected the buttons;









LED + buttons + LCD (2)

- interruption vs. pooling:
  - analysis of the TEXAS Manual for the ports, timers and LCD controller registers for interruptions;

Port	Register	Short Form	Address	Register Type	Initial State
P1	Input	P1IN	020h	Read only	–
	Output	P1OUT	021h	Read/write	Unchanged
	Direction	P1DIR	022h	Read/write	Reset with PUC
	Interrupt Flag	P1IFG	023h	Read/write	Reset with PUC
	Interrupt Edge Select	P1IES	024h	Read/write	Unchanged
	Interrupt Enable	P1IE	025h	Read/write	Reset with PUC
	Port Select	P1SEL	026h	Read/write	Reset with PUC
P2	Input	P2IN	028h	Read only	–
	Output	P2OUT	029h	Read/write	Unchanged
	Direction	P2DIR	02Ah	Read/write	Reset with PUC
	Interrupt Flag	P2IFG	02Bh	Read/write	Reset with PUC
	Interrupt Edge Select	P2IES	02Ch	Read/write	Unchanged
	Interrupt Enable	P2IE	02Dh	Read/write	Reset with PUC
	Port Select	P2SEL	02Eh	Read/write	Reset with PUC

IE2, Interrupt Enable Register 2

7	6	5	4	3	2	1	0
BTIE							
rw-0							

BTIE Bit 7 Basic Timer1 interrupt enable. This bit enables the BTIFG interrupt Because other bits in IE2 may be used for other modules, it is recommended to set or clear this bit using `BIS.B` or `BIC.B` instructions, rather than `MOV.B` or `CLR.B` instructions.
0 Interrupt not enabled
1 Interrupt enabled

IFG2, Interrupt Flag Register 2

7	6	5	4	3	2	1	0
BTIFG							
rw-0							

BTIFG Bit 7 Basic Timer1 interrupt flag. Because other bits in IFG2 may be used for other modules, it is recommended to clear BTIFG automatically by servicing the interrupt, or by using `BIS.B` or `BIC.B` instructions, rather than `MOV.B` or `CLR.B` instructions.
0 No interrupt pending
1 Interrupt pending

Bits 6-1 These bits may be used by other modules. See device-specific datasheet.

- hands-on Lab.

Buzzer (1)

- timers description: Timer_A and Timer_B;
- search in the TEXAS Manual for the chapter related Timer_A and Timer_B characteristics:
  - explanation of timers modes of operation;
  - explanation of timers capture/compare blocks: output units;
  - presentation of timers configuration: registers.

Chapter 12

Timer_A

Timer_A is a 16-bit timer/counter with multiple capture/compare registers. This chapter describes Timer_A. Timer_A3 (three capture/compare registers) is implemented in all MSP430x4xx devices. Timer1_A5 (five capture/compare registers) is also implemented on MSP430x415, MSP430x417, and MSP430xW42x devices.

Topic	Page
12.1 Timer_A Introduction	12-2
12.2 Timer_A Operation	12-4
12.3 Timer_A Registers	12-19

Table 12–1. Timer Modes

MCx	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TACCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TACCR0 and back down to zero.

Figure 12–11. Capture Cycle

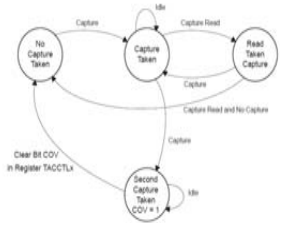
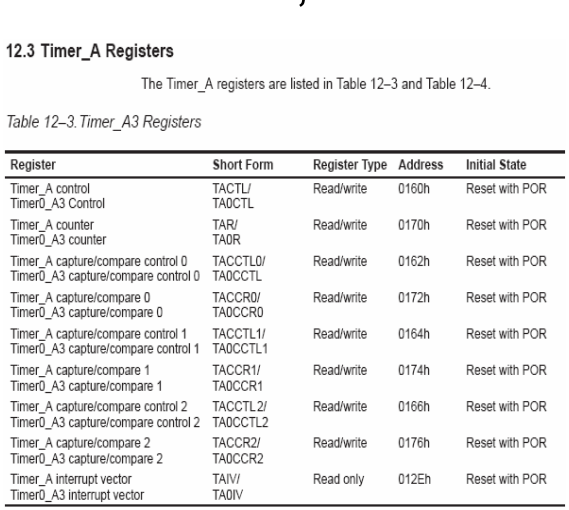


Table 12–2. Output Modes

OUTMODx	Mode	Description
000	Output	The output signal OUTx is defined by the OUTx bit. The OUTx signal updates immediately when OUTx is updated.
001	Set	The output is set when the timer *counts* to the TACCRx value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer *counts* to the TACCRx value. It is reset when the timer *counts* to the TACCR0 value.
011	Set/Reset	The output is set when the timer *counts* to the TACCRx value. It is reset when the timer *counts* to the TACCR0 value.
100	Toggle	The output is toggled when the timer *counts* to the TACCRx value. The output period is double the timer period.
101	Reset	The output is reset when the timer *counts* to the TACCRx value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer *counts* to the TACCRx value. It is set when the timer *counts* to the TACCR0 value.
111	Reset/Set	The output is reset when the timer *counts* to the TACCRx value. It is set when the timer *counts* to the TACCR0 value.

Buzzer (2)

- search in the IAR Systems Manual for the syntax to create interruptions for the task (also look in MSPGCC and Code Composer Essentials manuals):
  - C programming specifications (particularities from C standard) to create a PWM (Pulse Width Modulation) operation to activate the buzzer;

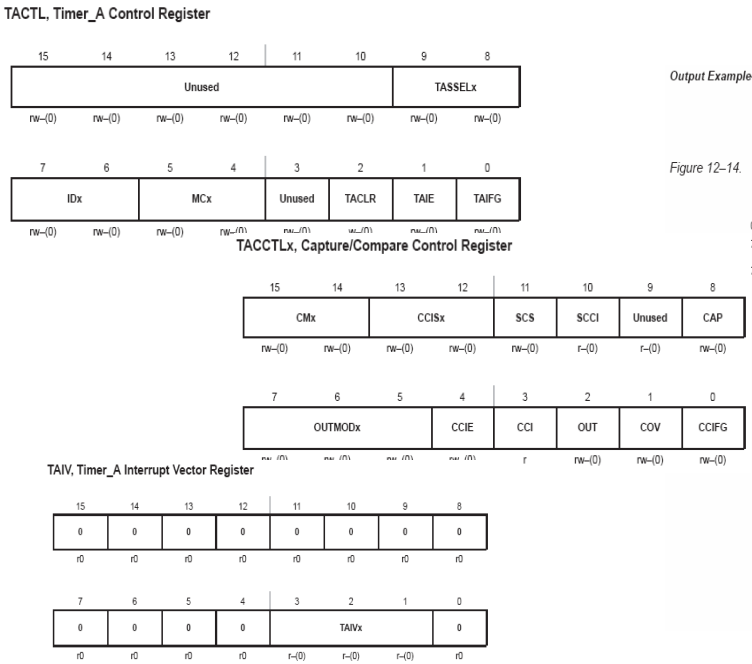


12.3 Timer_A Registers

The Timer_A registers are listed in Table 12–3 and Table 12–4.

Table 12–3. Timer_A3 Registers

Register	Short Form	Register Type	Address	Initial State
Timer_A control Timer0_A3 Control	TACTL/ TA0CTL	Read/write	0160h	Reset with POR
Timer_A counter Timer0_A3 counter	TAR/ TA0R	Read/write	0170h	Reset with POR
Timer_A capture/compare control 0 Timer0_A3 capture/compare control 0	TACCTL0/ TA0CCTL	Read/write	0162h	Reset with POR
Timer_A capture/compare 0 Timer0_A3 capture/compare 0	TACCR0/ TA0CCR0	Read/write	0172h	Reset with POR
Timer_A capture/compare control 1 Timer0_A3 capture/compare control 1	TACCTL1/ TA0CCTL1	Read/write	0164h	Reset with POR
Timer_A capture/compare 1 Timer0_A3 capture/compare 1	TACCR1/ TA0CCR1	Read/write	0174h	Reset with POR
Timer_A capture/compare control 2 Timer0_A3 capture/compare control 2	TACCTL2/ TA0CCTL2	Read/write	0166h	Reset with POR
Timer_A capture/compare 2 Timer0_A3 capture/compare 2	TACCR2/ TA0CCR2	Read/write	0176h	Reset with POR
Timer_A interrupt vector Timer0_A3 interrupt vector	TAIV/ TA0IV	Read only	012Eh	Reset with POR



TACTL, Timer_A Control Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

TACCTLx, Capture/Compare Control Register

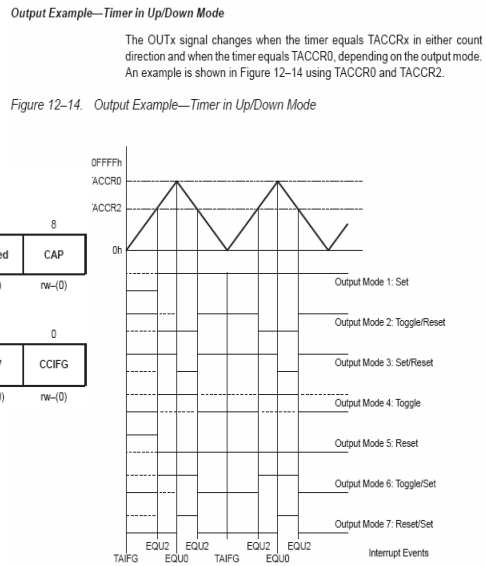
15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	r–(0)	r–(0)	rw–(0)

7	6	5	4	3	2	1	0
OUTMODx				CCIE	CCI	OUT	COV	CCIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	r	rw–(0)	rw–(0)	rw–(0)

TAIV, Timer_A Interrupt Vector Register

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	0	0	TAIVx			0
r0	r0	r0	r0	r–(0)	r–(0)	r–(0)	r0



Output Example—Timer in Up/Down Mode

The OUTx signal changes when the timer equals TACCRx in either count direction and when the timer equals TACCR0, depending on the output mode. An example is shown in Figure 12–14 using TACCR0 and TACCR2.

Figure 12–14. Output Example—Timer in Up/Down Mode

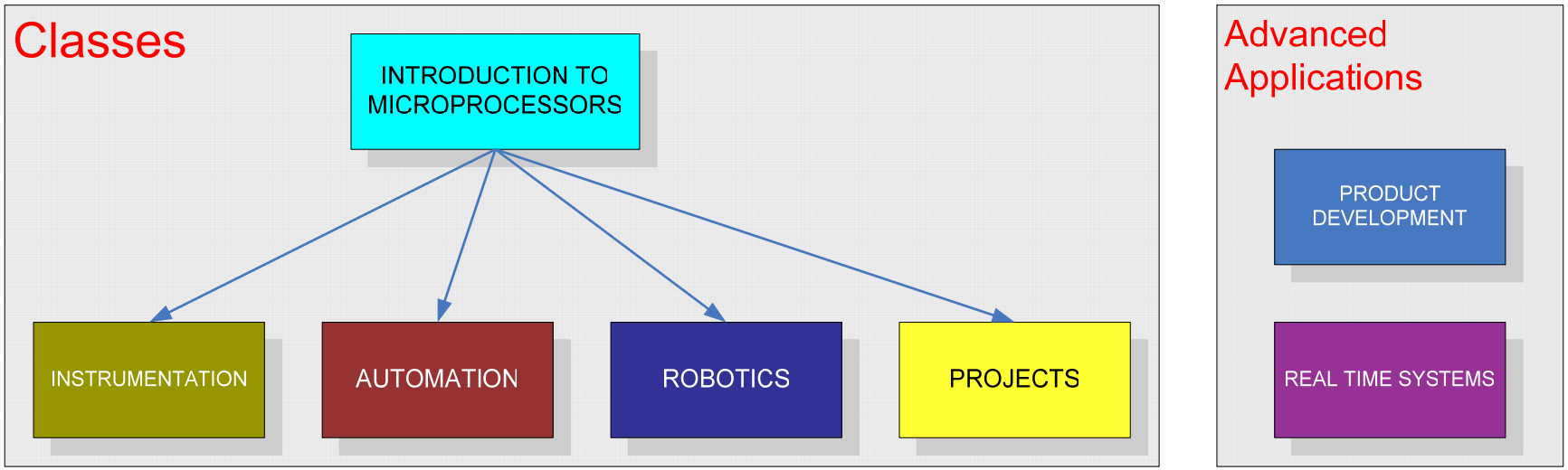
- hands-on Lab.

LCD Clock with OLIMEX kit

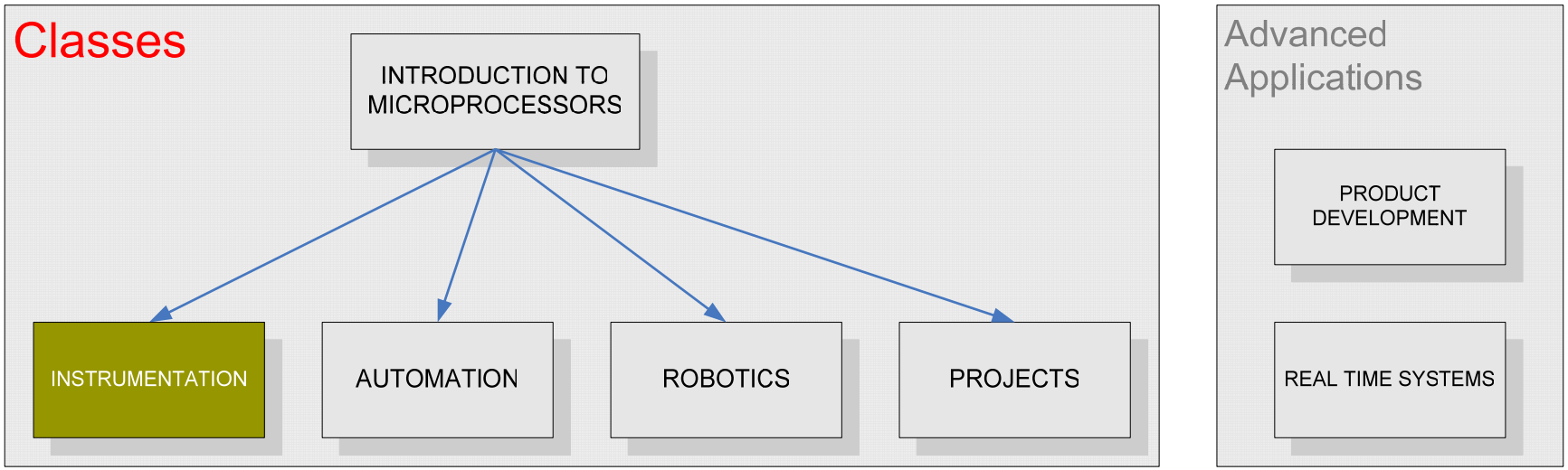
- development task: create a clock to work in ***hh:mm*** or ***mm:ss*** by buttons selection;
- analysis of the OLIMEX kit schema with MSP430F449 (STK and STK2);
- ports registers configuration: LED and buttons;
- LCD controller configuration: registers;
- Basic Timer and Watchdog Timer configuration;
- interruptions definition;

- hands-on Lab.

MSP430@UBI

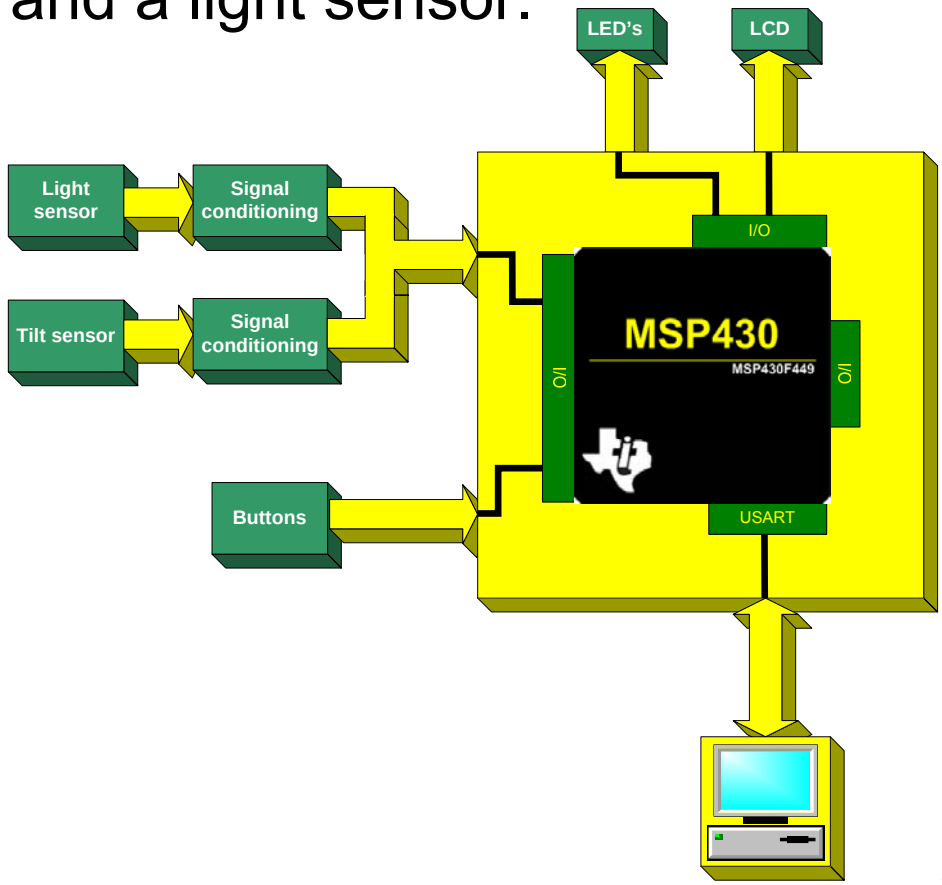

Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

Instrumentation



MSP430 Monitoring System

Objective: develop a low cost system for event record. System must save on memory time stamps of events detected by a tilt sensor and a light sensor.



MSP430 Monitoring System

Implementation tasks:

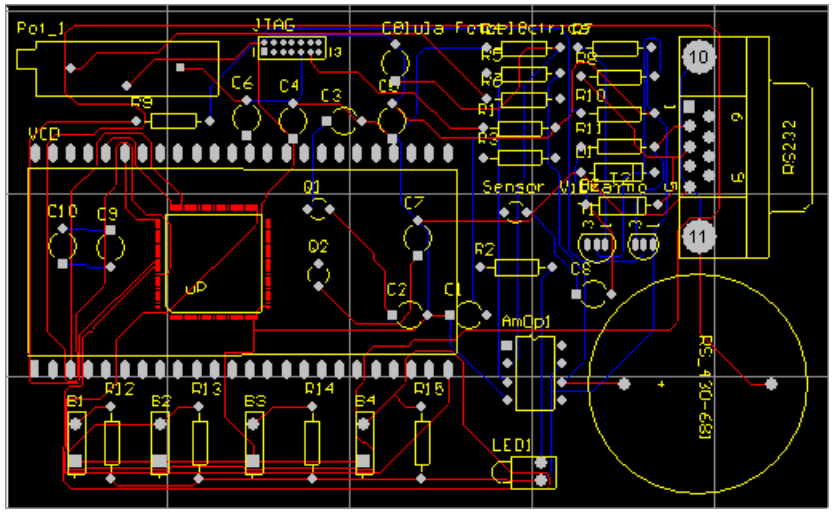
- two audits modes {Manual and Automatic};
- develop sensors interface electronics;
- buttons and LCD to user interface;
- implement a software real time clock;
- PC interface.

Recommended MSP430 resources:

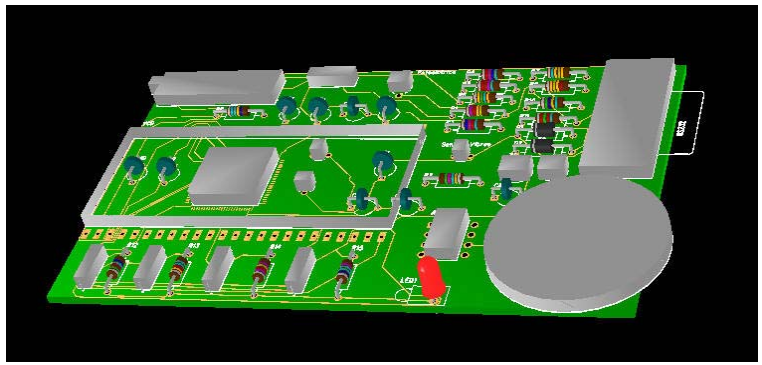
- I/O ports for LED, LCD and buttons on board interface;
- Basic Timer;
- USART.

MSP430 Monitoring System

Implementation: Hardware developed board



PCB board routing

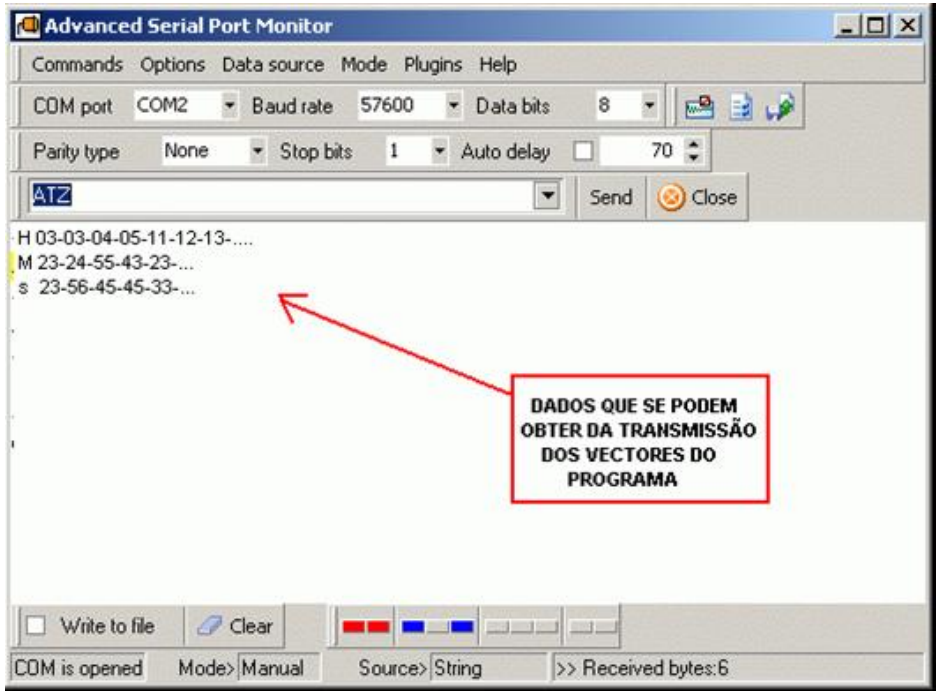


Board 3D view

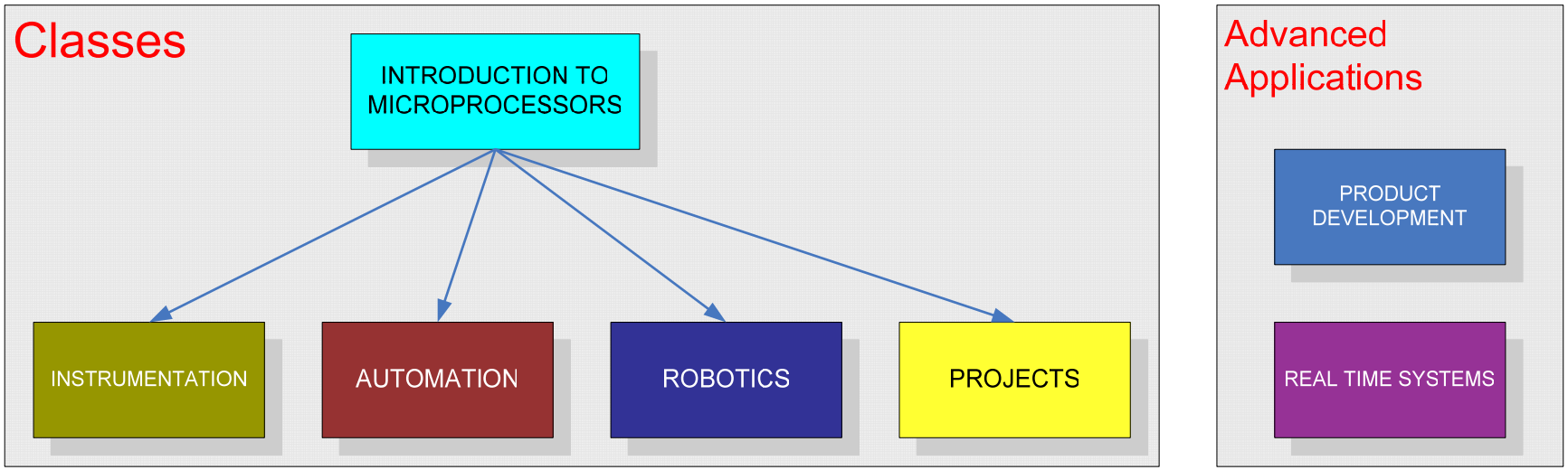
Instrumentation

MSP430 Monitoring System

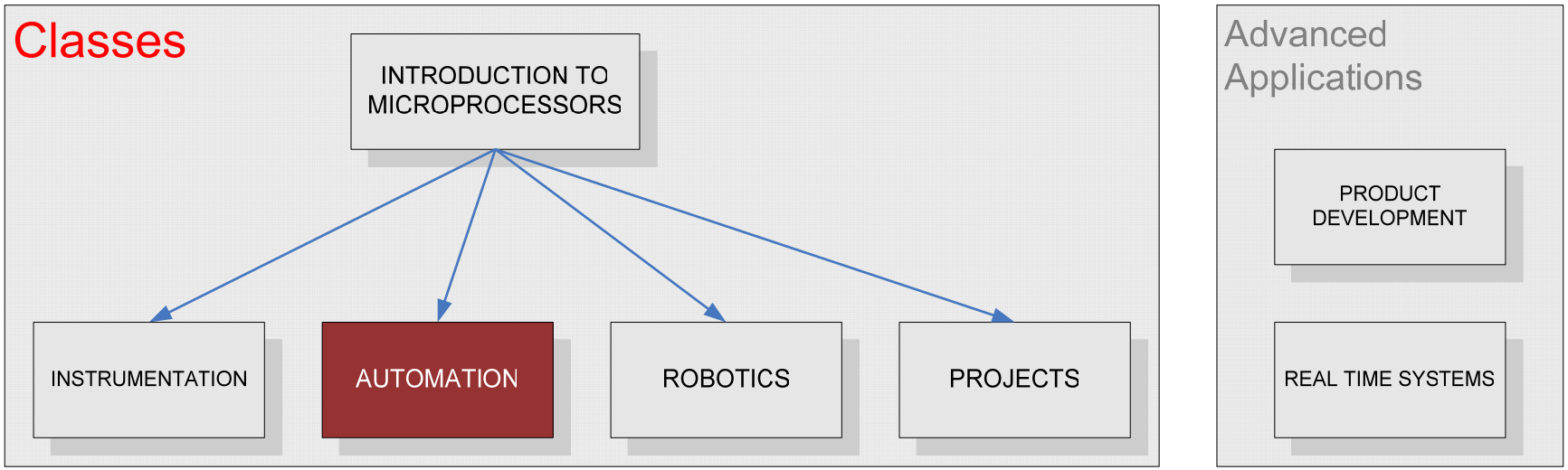
Implementation: PC data transmission example



MSP430@UBI


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

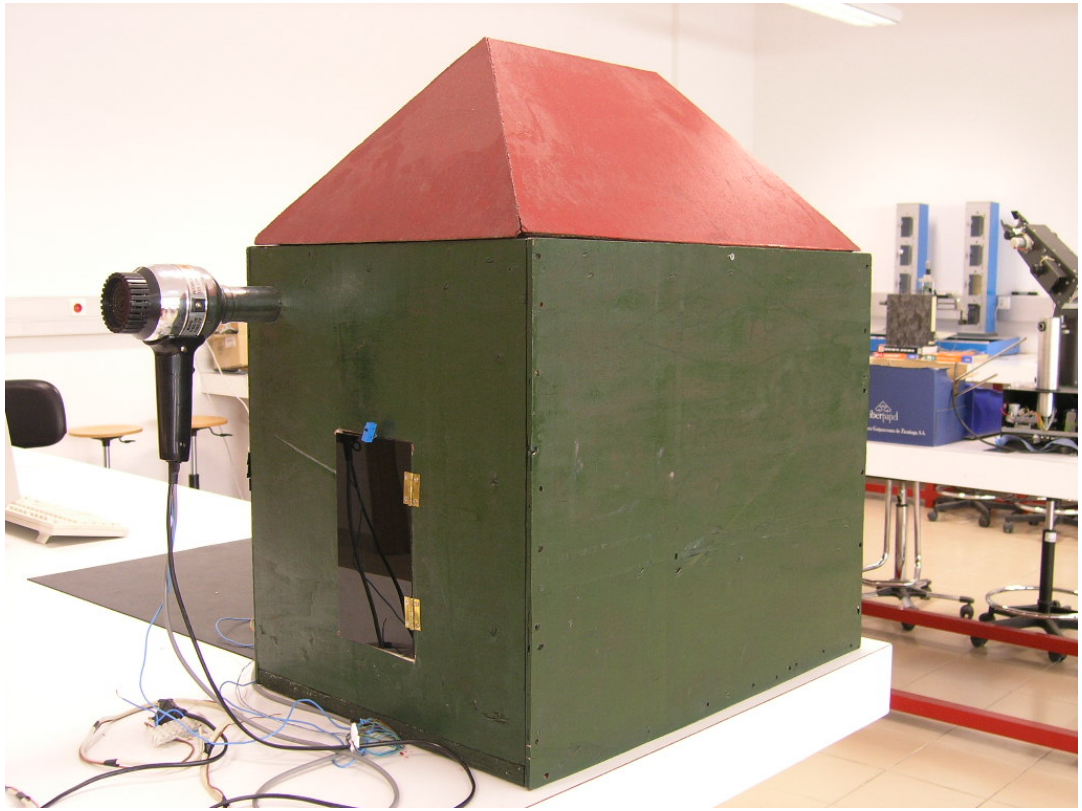
Automation


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

Home Automation System

Objective:

MSP430 home heating, illumination and security control.



Home Automation System

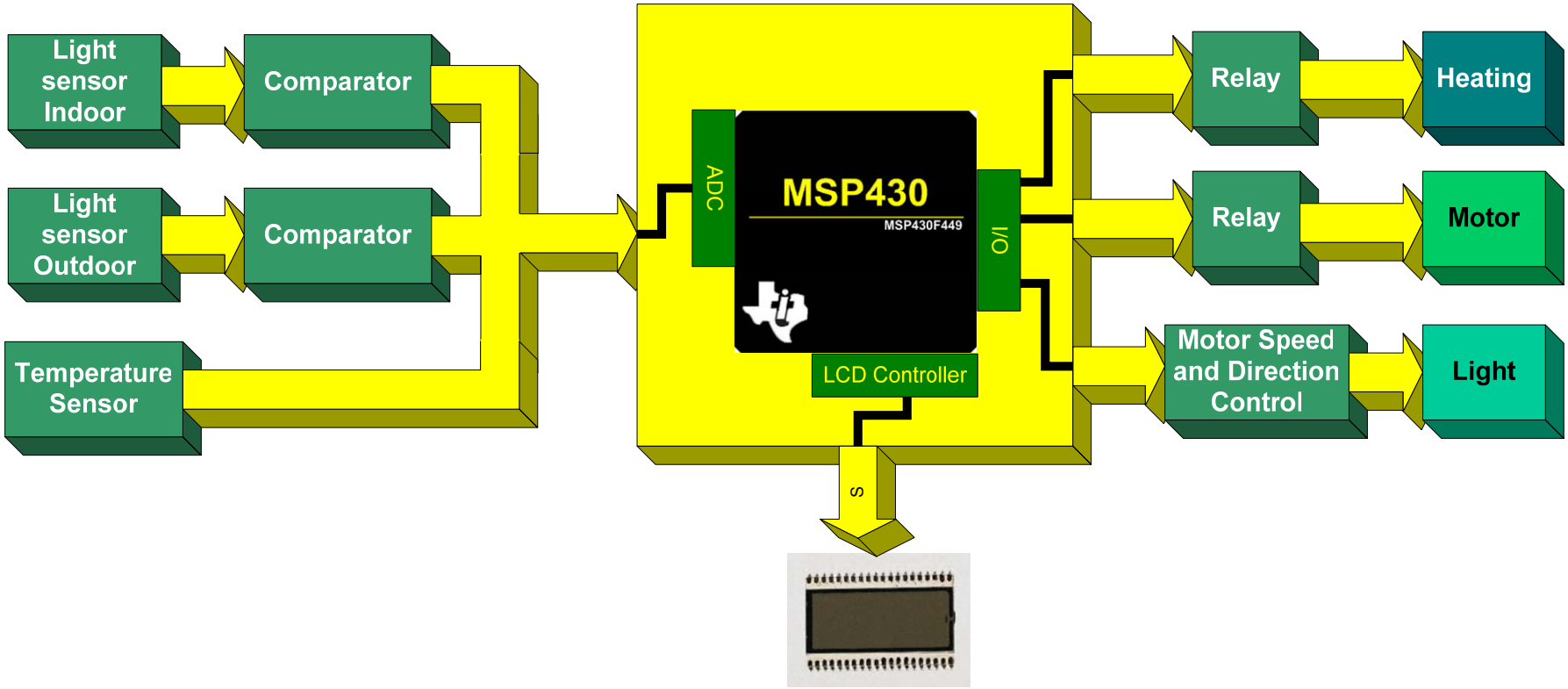
Implementation Tasks:

- clock;
- indication of actual temperature;
- possibility of two daily temperature changes;
- use of LCD kit for easy user interface;
- register of all home events.

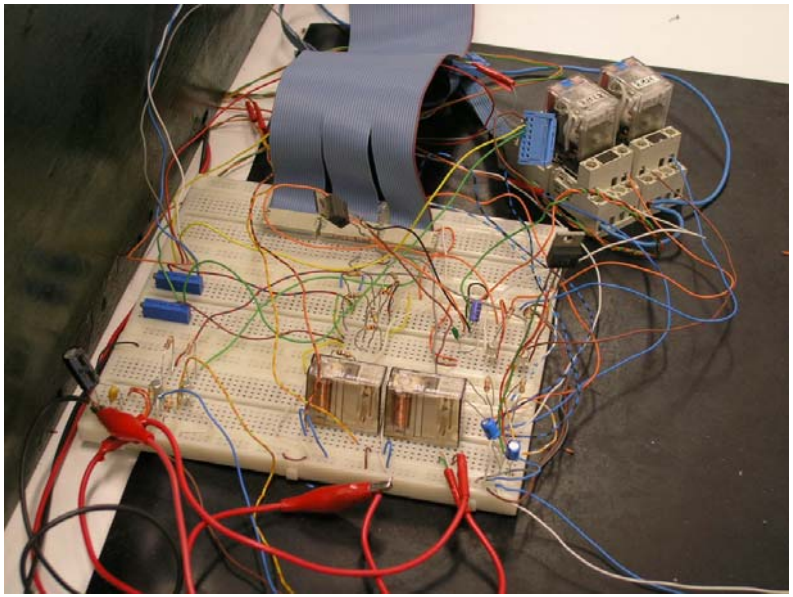
Recommended MSP430 resources:

- Digital I/O;
- LCD Controller;
- ADC12.

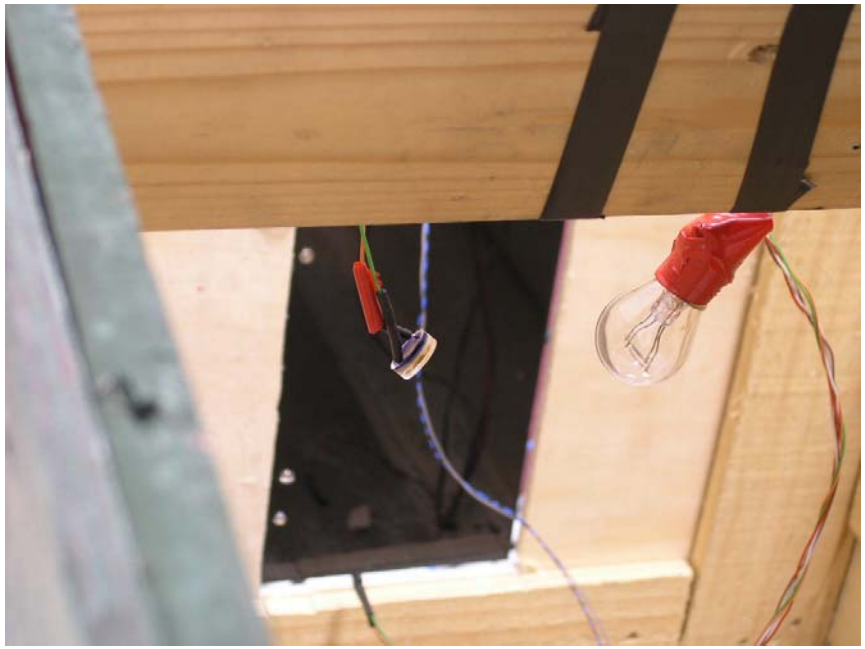
Home Automation System



Home Automation System

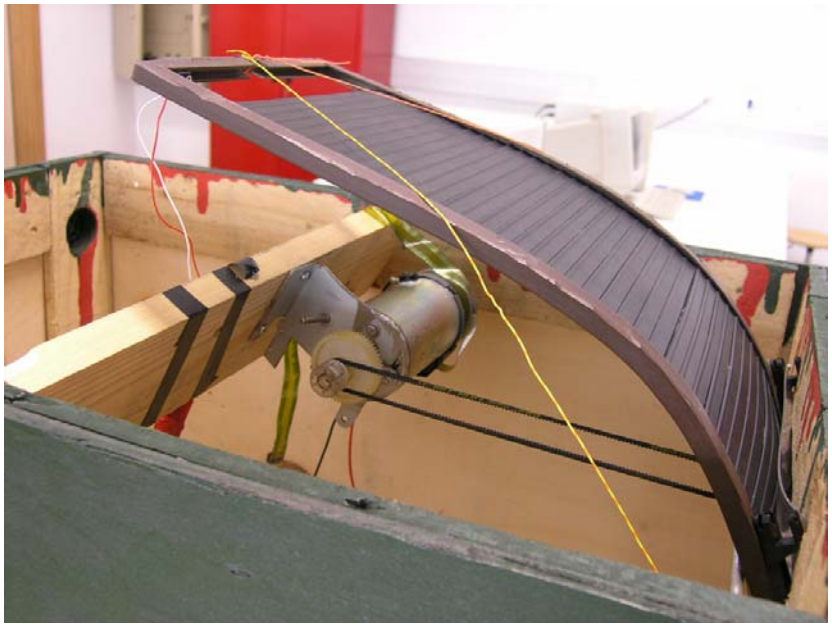


Development board

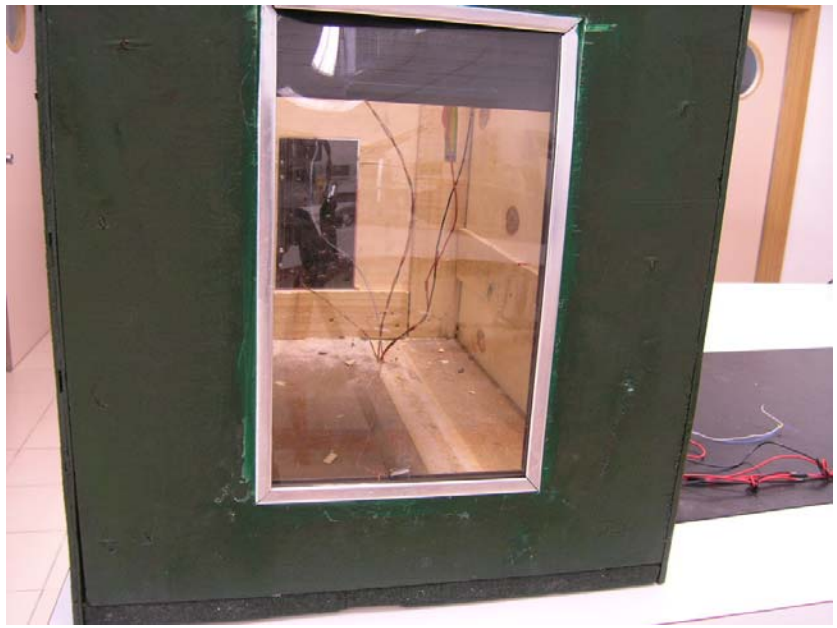


Indoor light sensor

Home Automation System

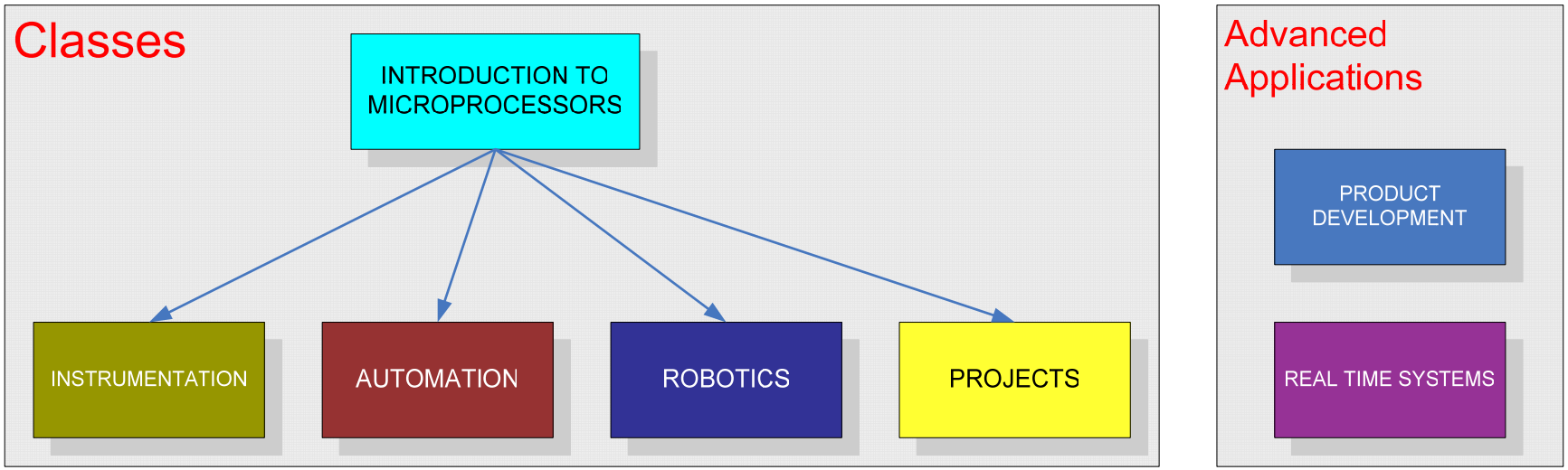


Window louver system

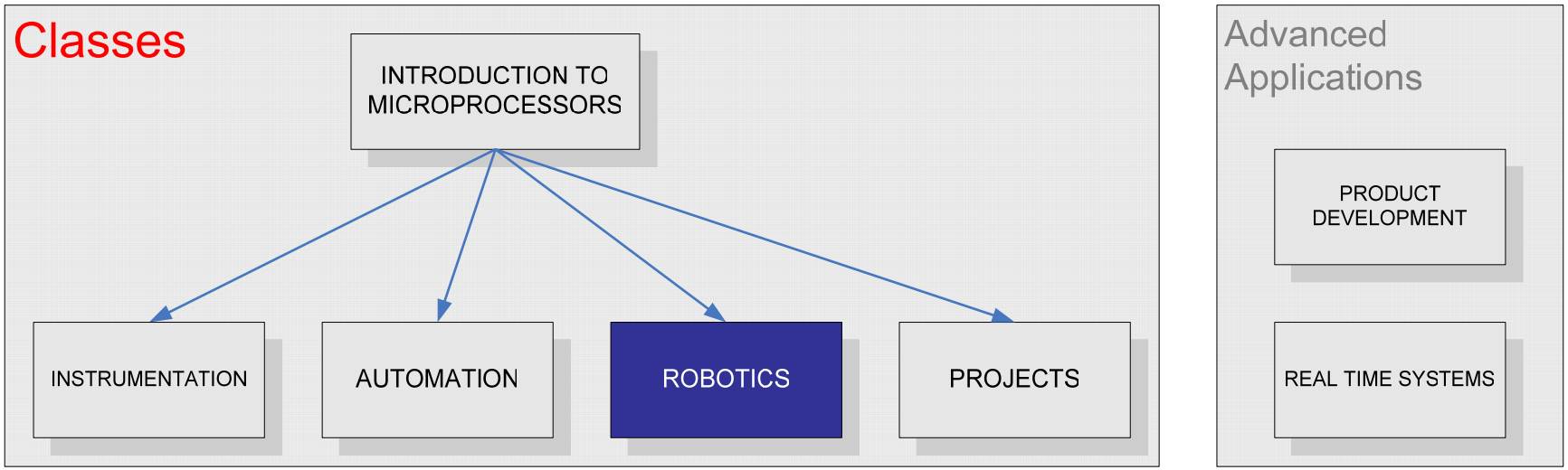


Window look

MSP430@UBI

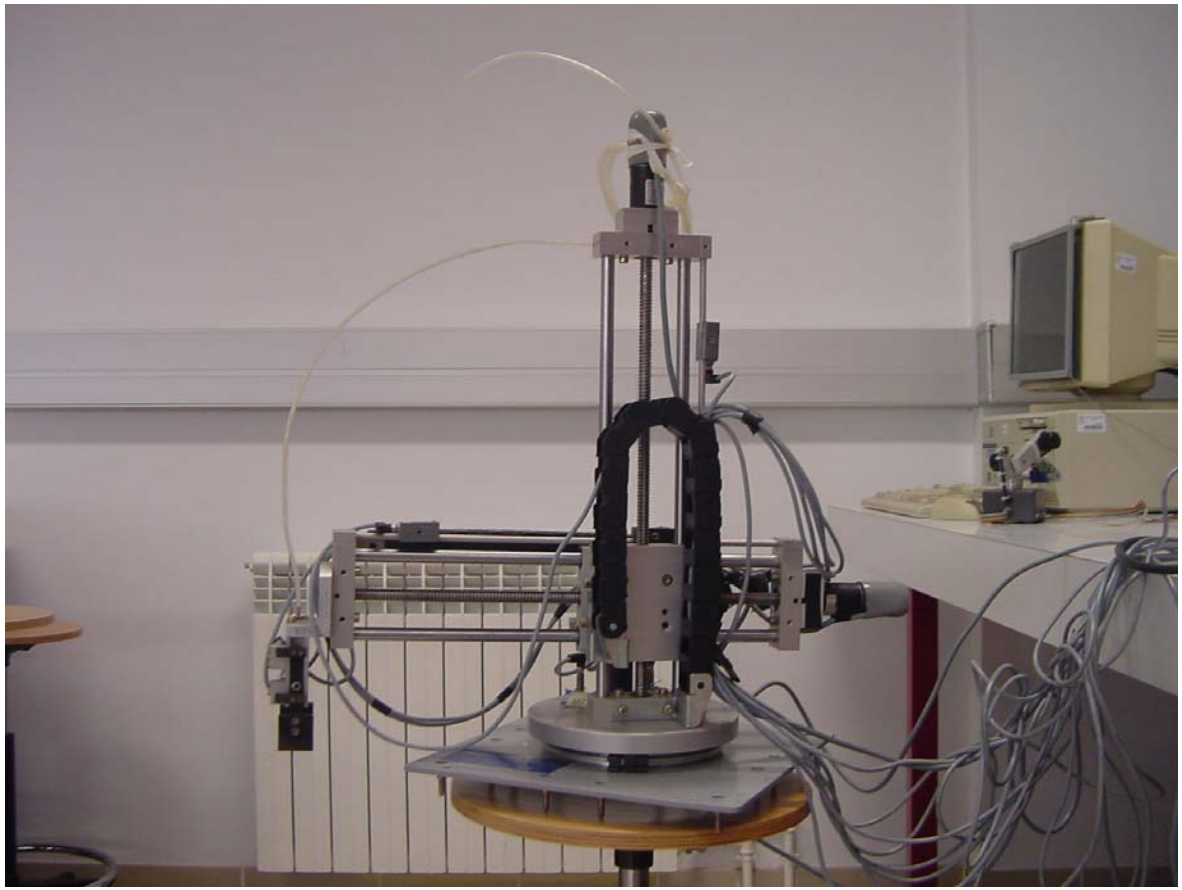

Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

Robotics


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

MSP430 Robot Implementation

Objective: Implementation of a control board and software for the FESTO robot.



MSP430 Robot Implementation

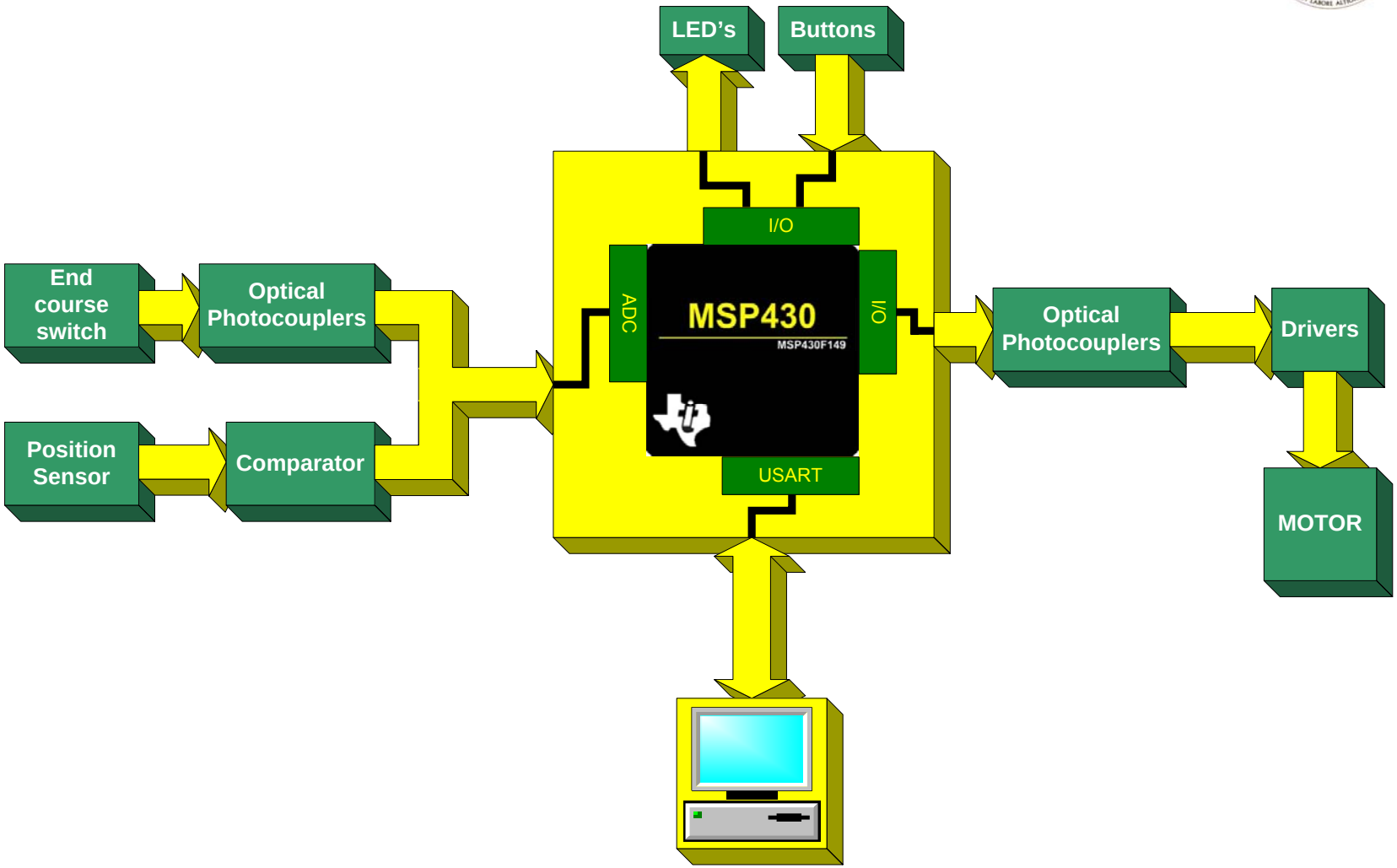
Implementation Tasks:

- connection to a PC via RS232;
- use all robot axis;
- possibility to save sequences for robot actions;
- use 16 digital Inputs/Outputs protected;
- load and save CAM instructions.

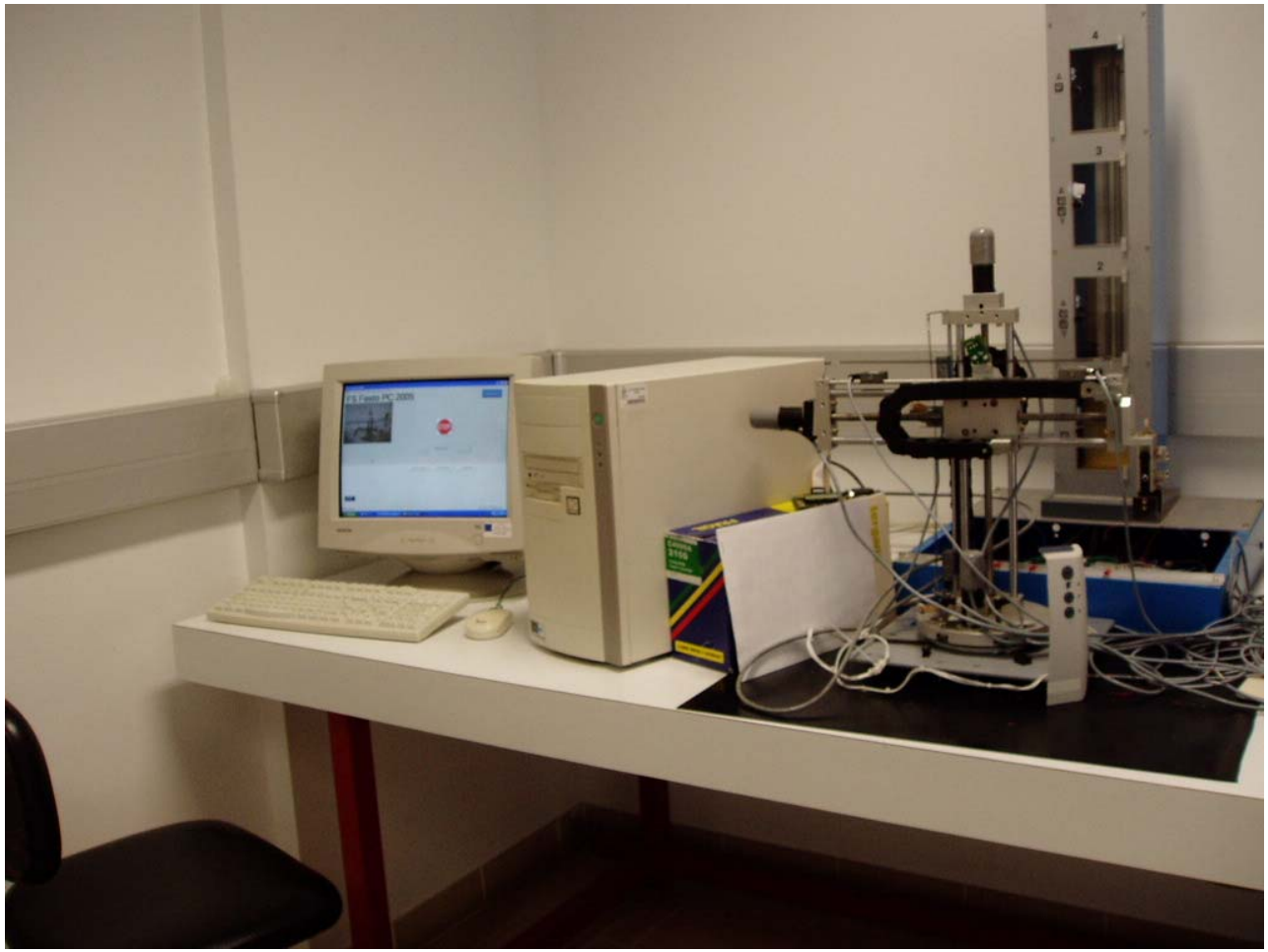
Recommended MSP430 Resources:

- USART;
- Digital I/O;
- ADC12.

MSP430 Robot Implementation

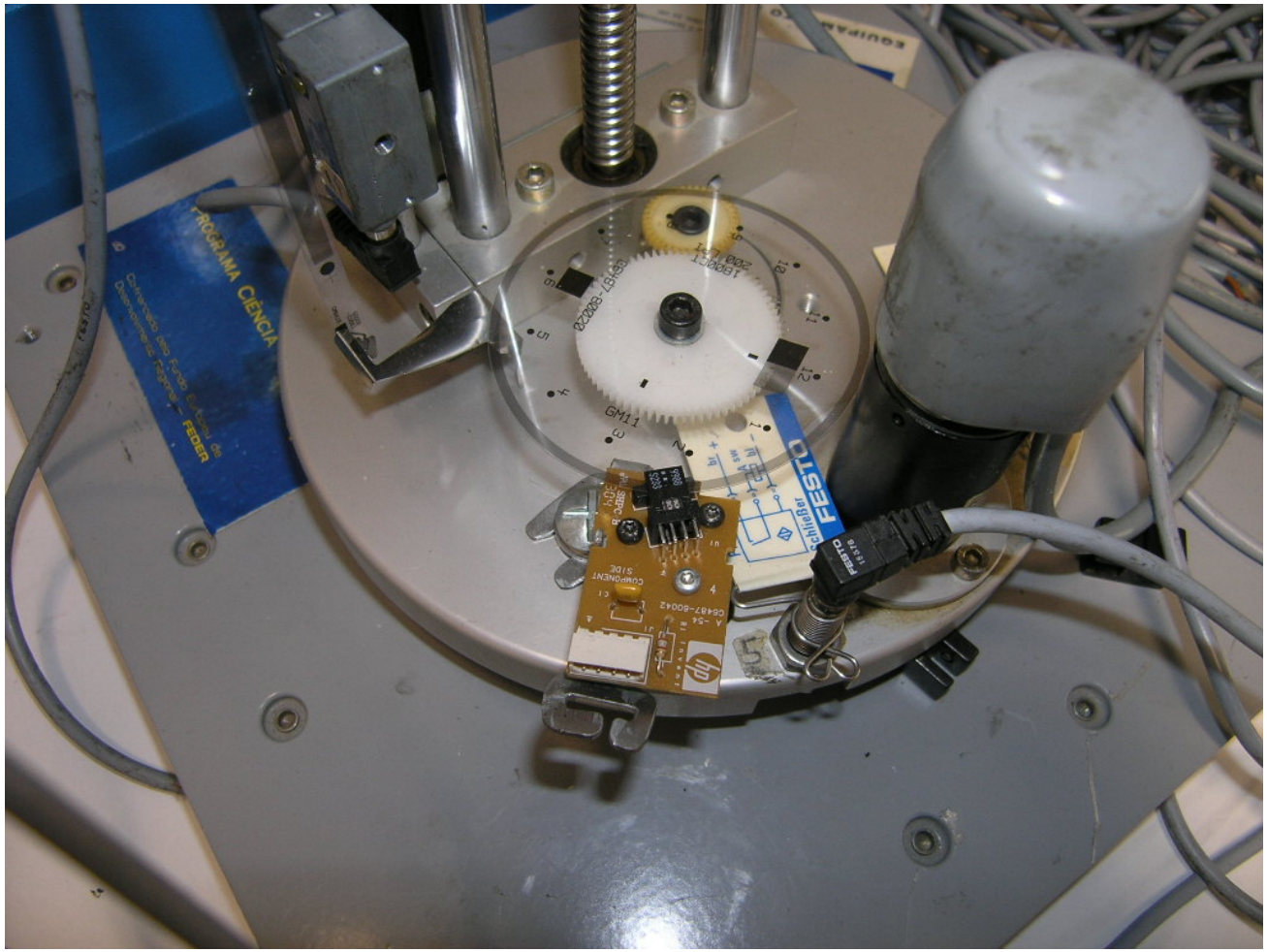


MSP430 Robot Implementation



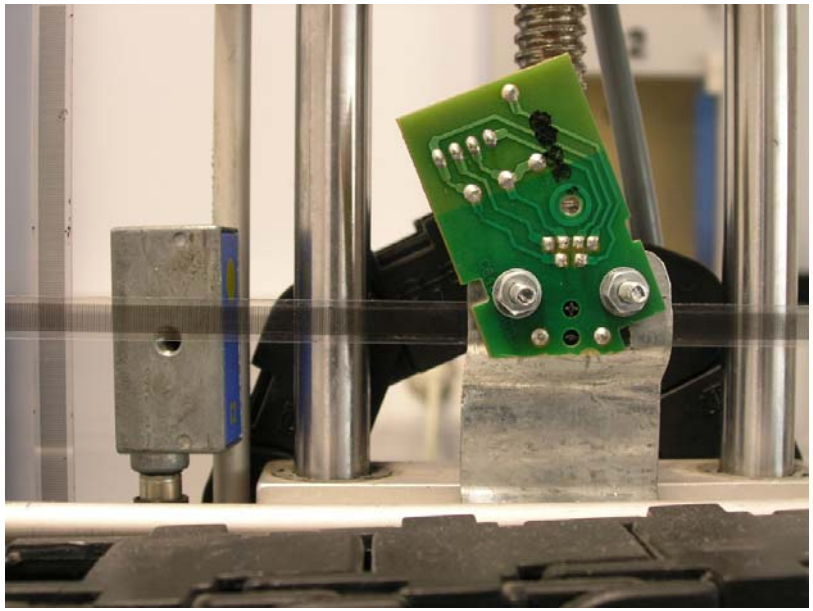
General view

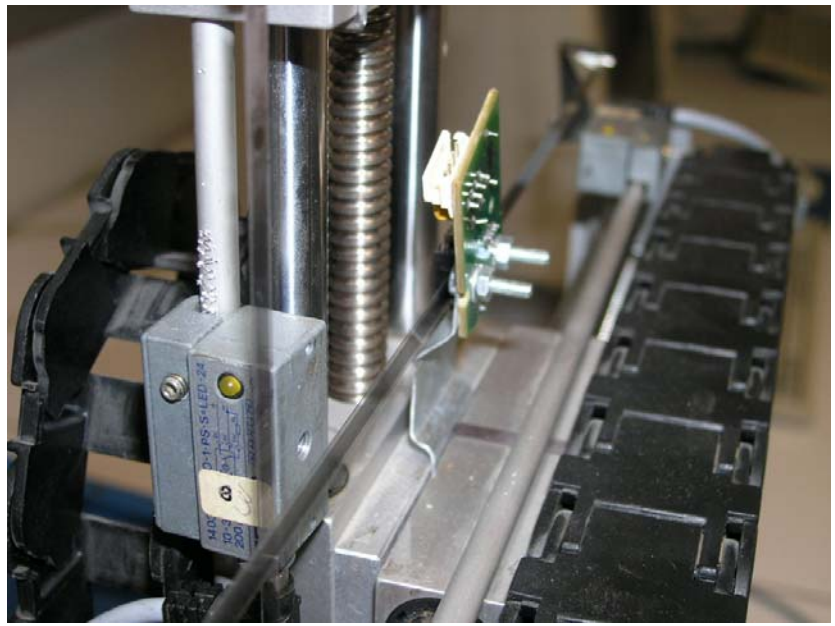
MSP430 Robot Implementation



Rotational optical sensor

MSP430 Robot Implementation



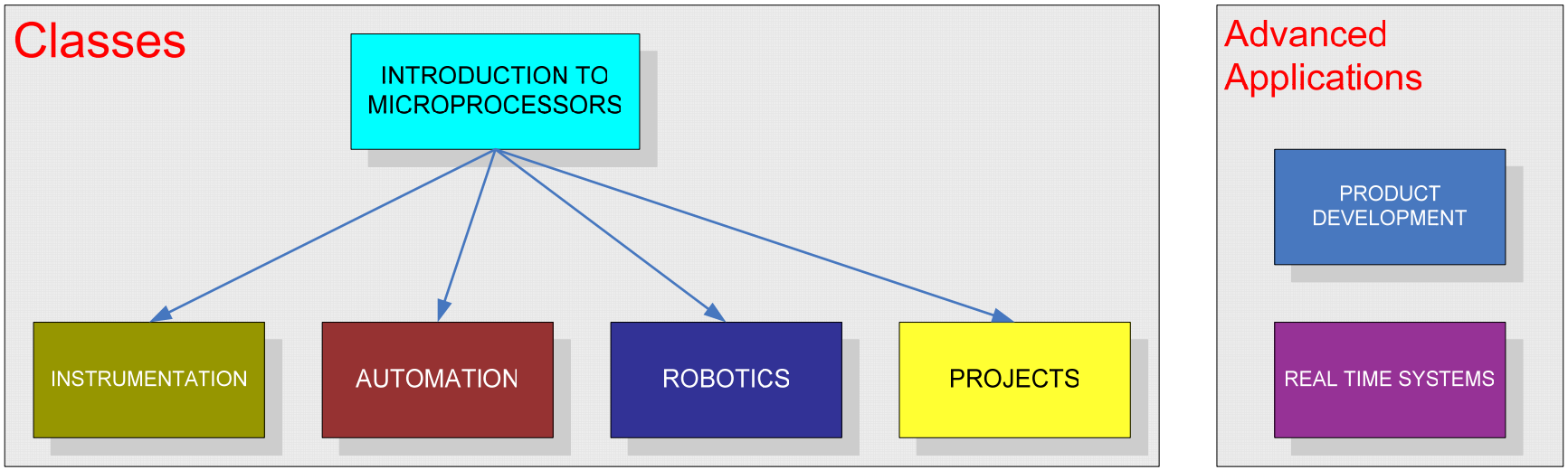


Linear optical sensor for longitudinal movement

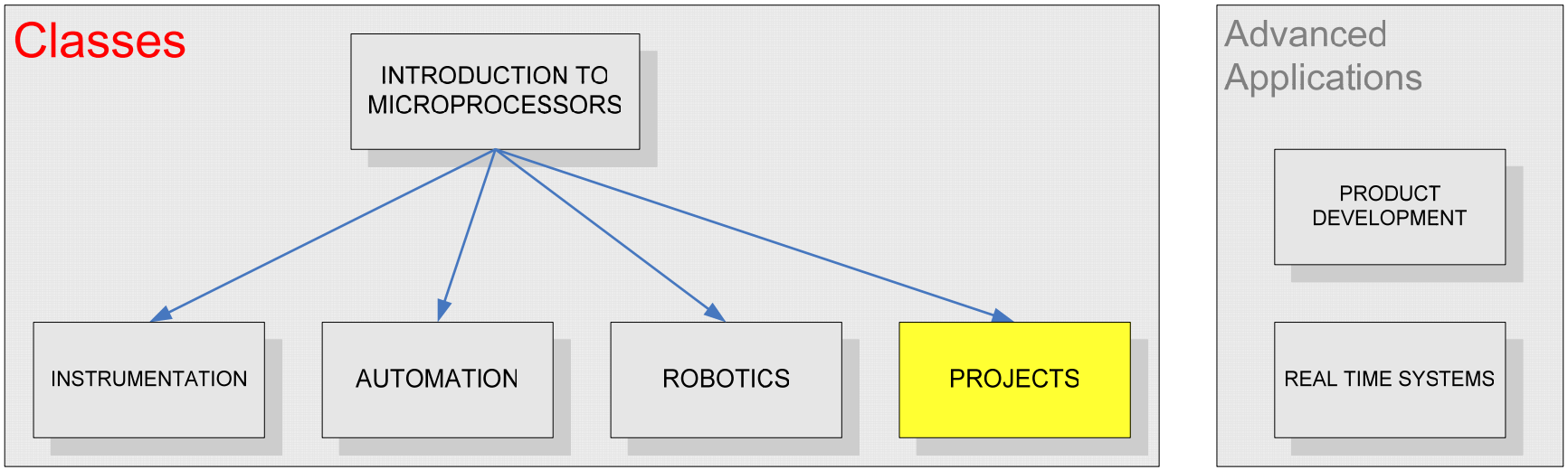
MSP430 Robot Implementation

Software to control the robot

MSP430@UBI


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

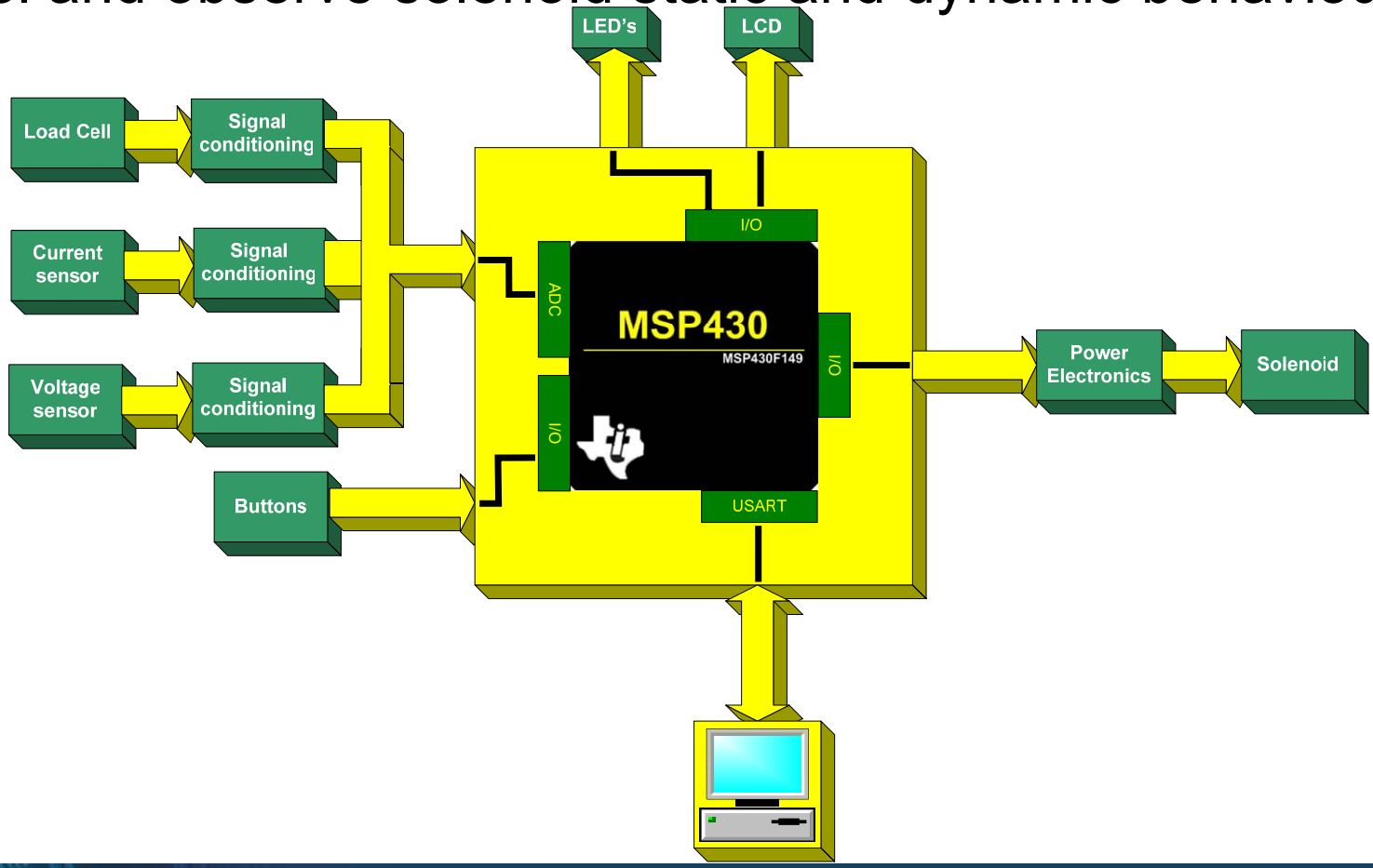
Projects


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

MSP430 Solenoid Experiment System

Objective:

Control and observe solenoid static and dynamic behaviour.



MSP430 Solenoid Experiment System

Implementation tasks:

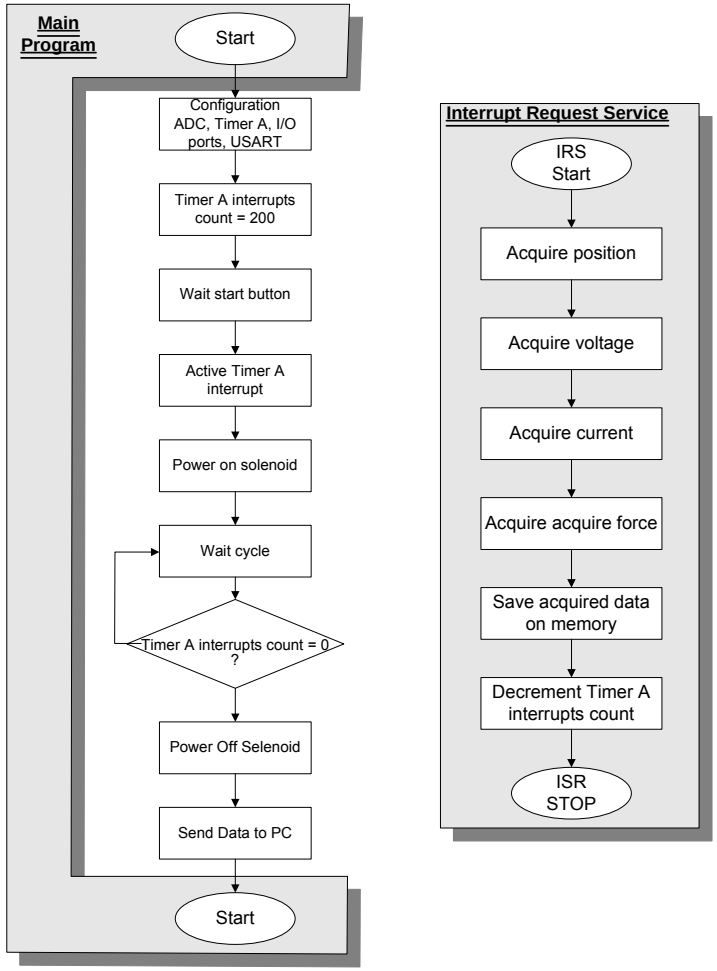
- in static experiment ⇒ acquire and process load cell information;
- in dynamic experiment ⇒ solenoid current, voltage and position acquisition;
- PC Interface in all experiments situations for data transmission;
- start/stop data acquisition and power feed.

Recommended MSP430 resources:

- I/O ports;
- on board buttons and LCD;
- Timer_A;
- ADC;
- interrupts;
- on board LED;
- on board RS232;
- USART.

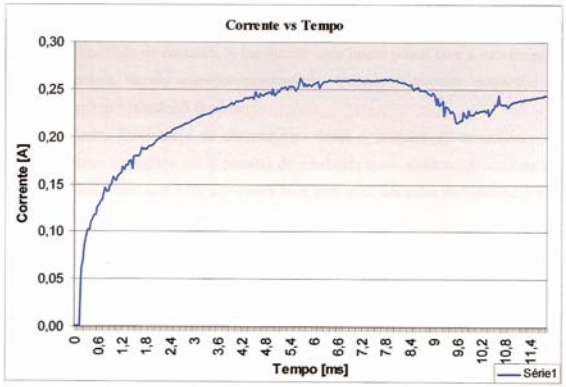
MSP430 Solenoid Experiment System

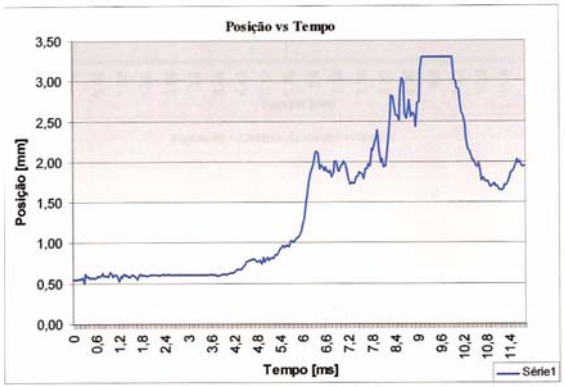
Implementation: software flowchart

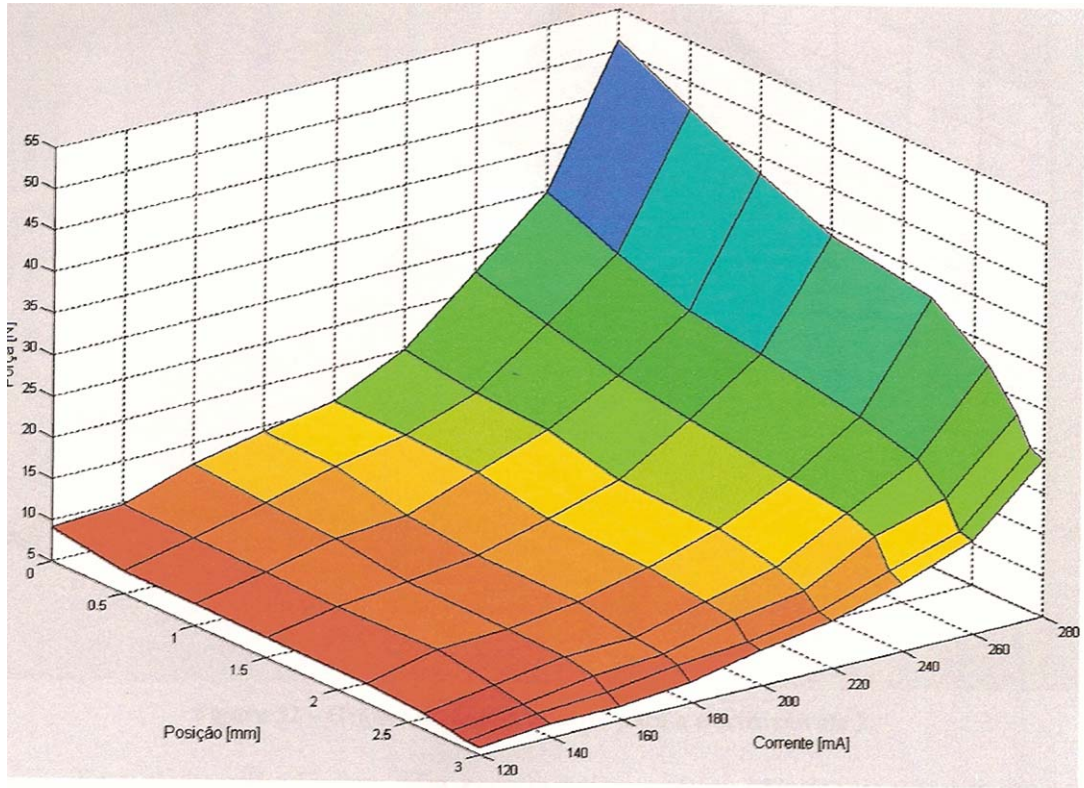


MSP430 Solenoid Experiment System

System results

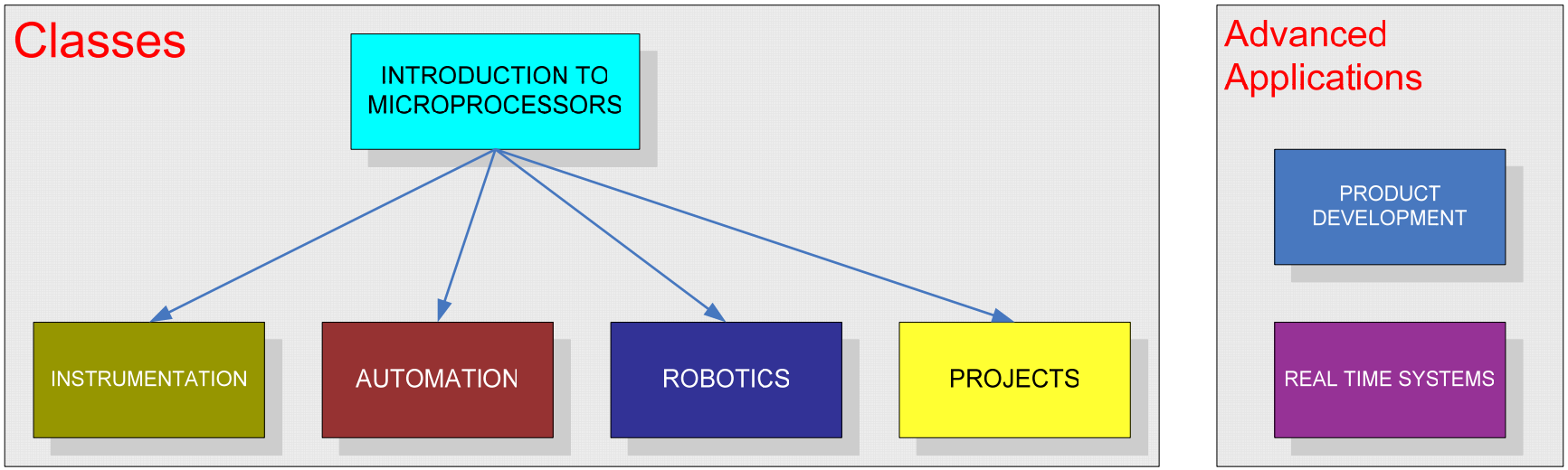




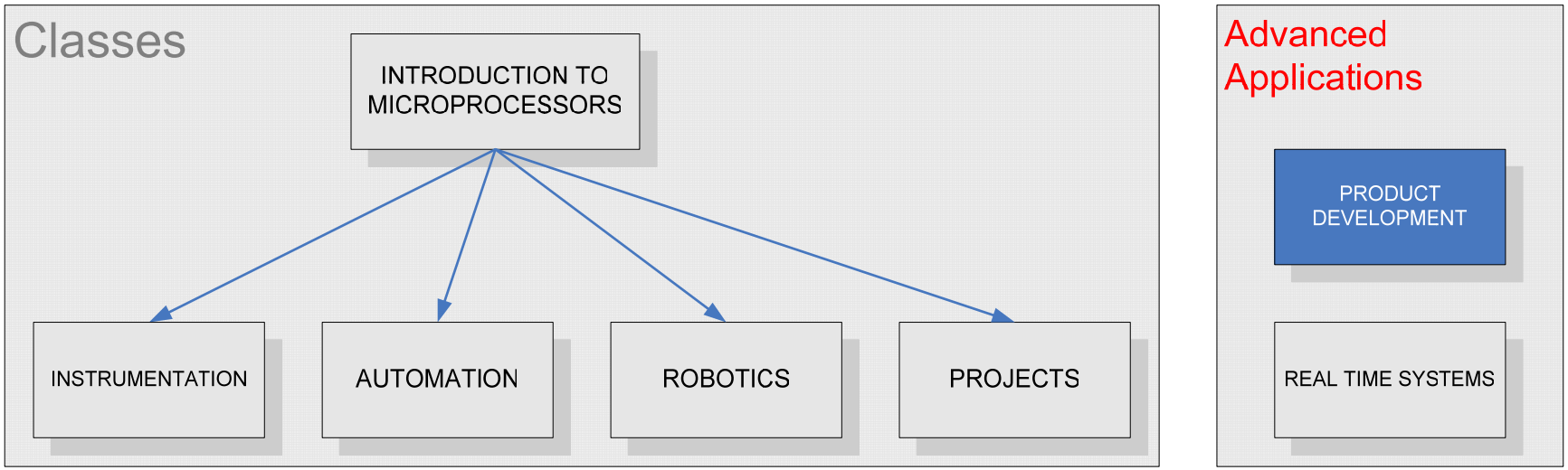


Projects

MSP430@UBI


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

Product Development

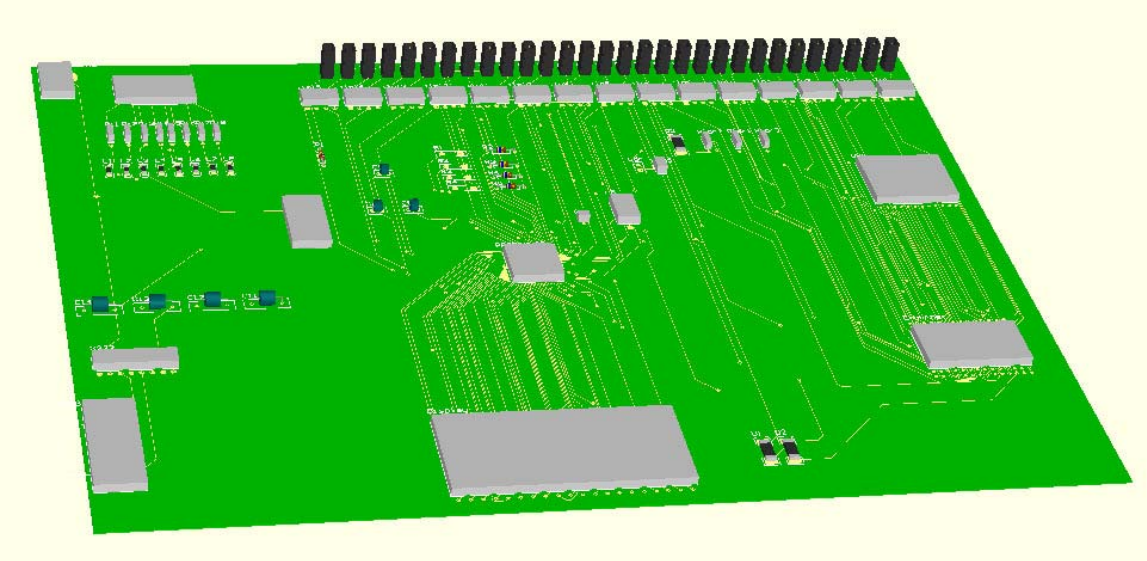


MSP430 advanced learning kit

Objective:

Development of a PCB board kit with several modules connected to one MSP430;

Learn how to draw MSP430 systems.



Product Development

MSP430 advanced learning kit

Implementation Tasks:

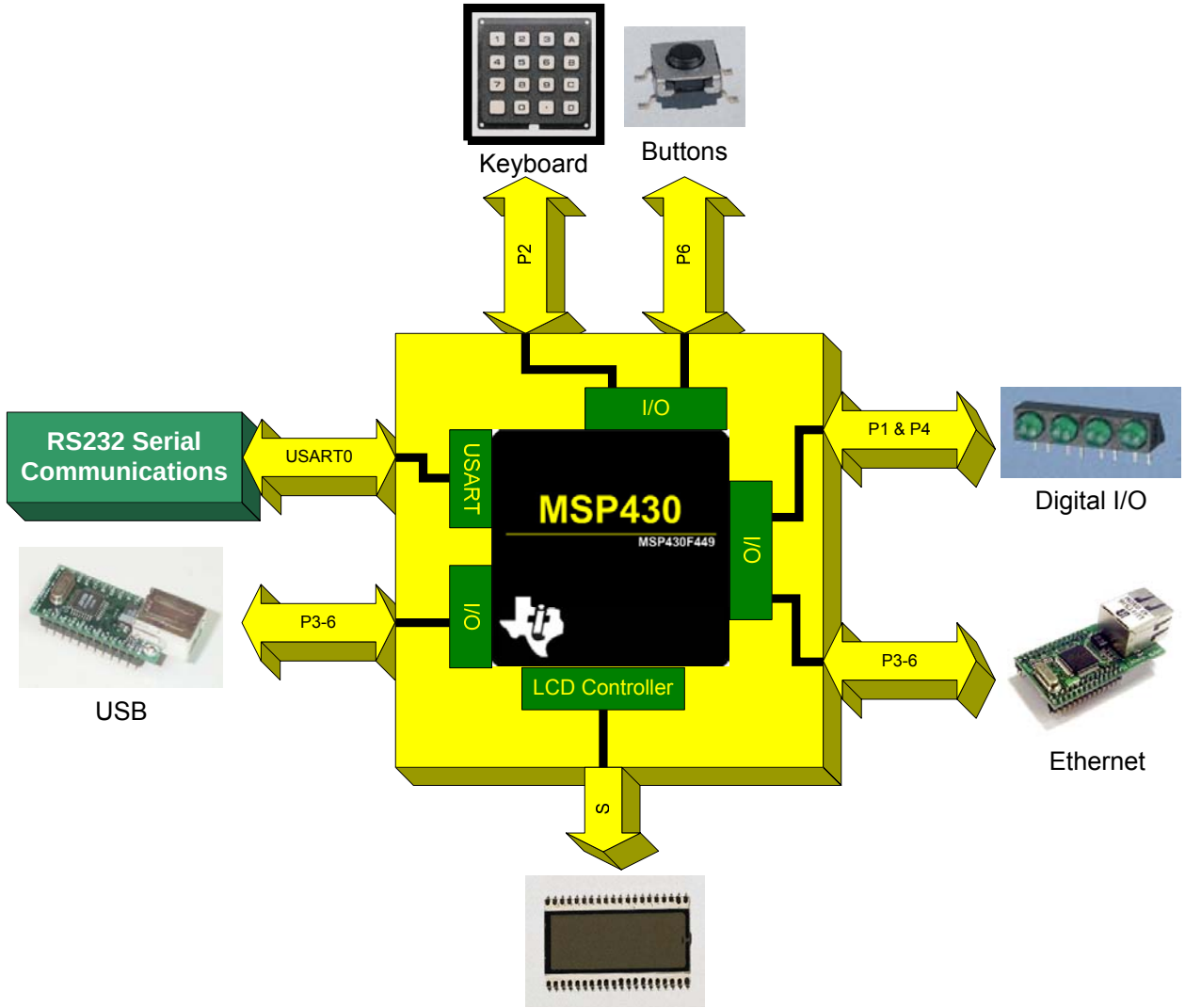
- possibility to connect and disconnect the modules;
- access to all microcontroller pins;
- several LED's connected to the ports;
- keyboard;
- on board LCD;
- Digital I/O protected (100 mA);
- Serial (RS232) communications;
- ethernet communications;
- temperature and luminosity sensor.

MSP430 advanced learning kit

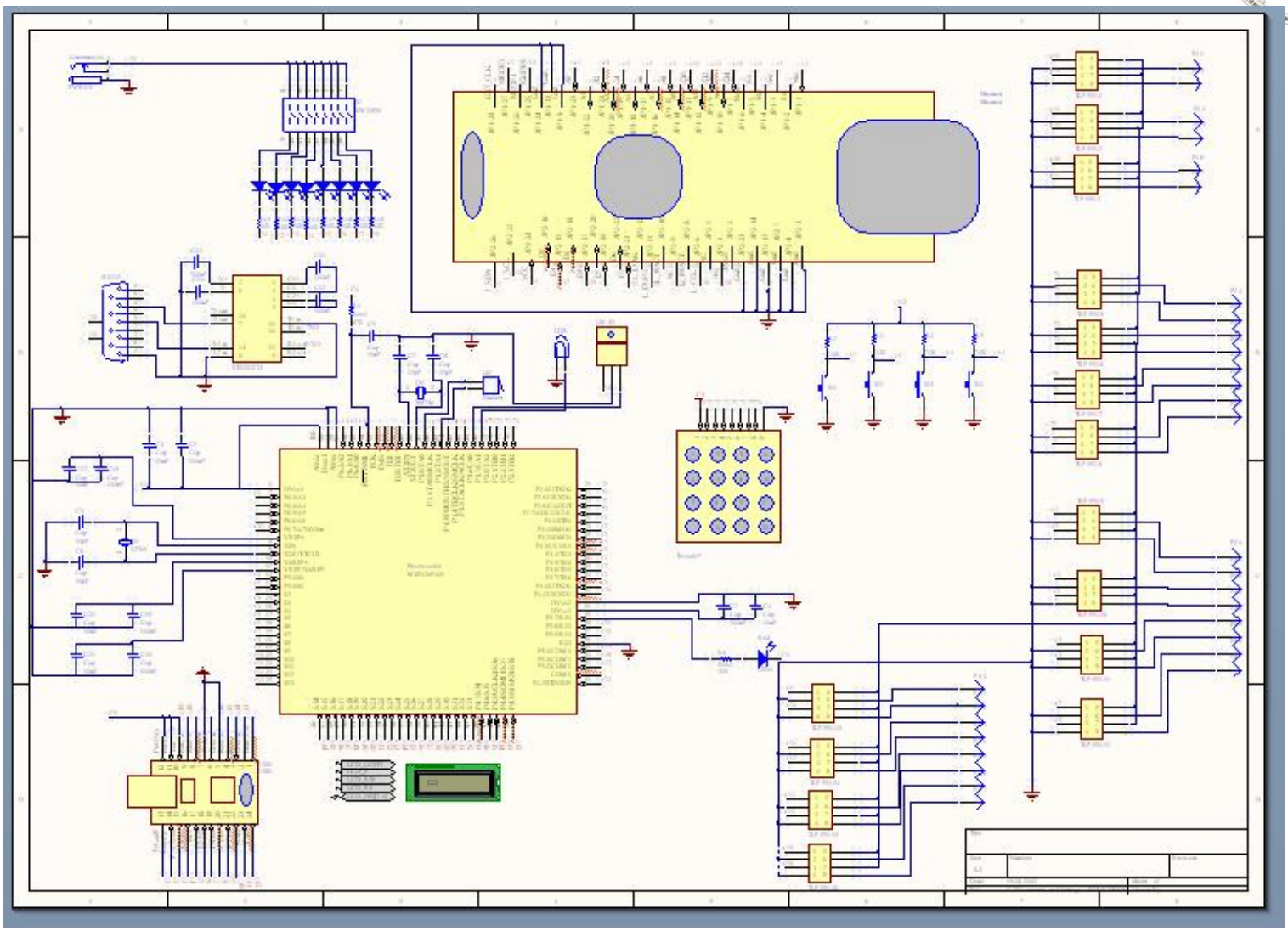
Recommended MSP430 Resources:

- Digital I/O;
- USART;
- LCD Controller;
- ADC12.

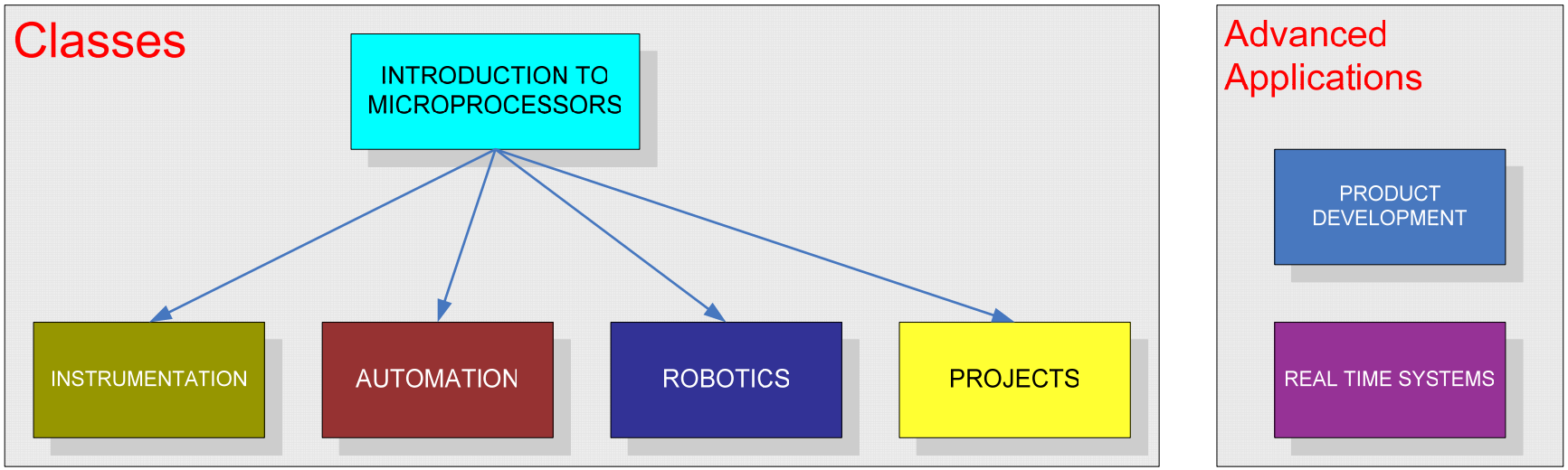
MSP430 advanced learning kit



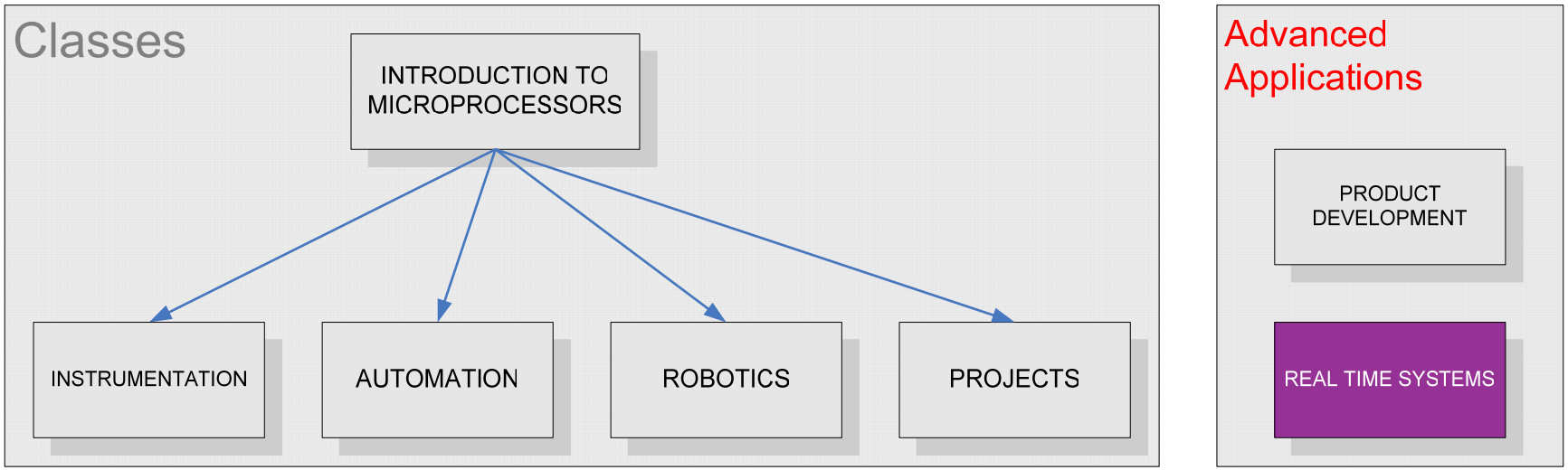
MSP430 advanced learning kit



MSP430@UBI


Classes
INTRODUCTION TO MICROPROCESSORS
INSTRUMENTATION
AUTOMATION
ROBOTICS
PROJECTS
Advanced Applications
PRODUCT DEVELOPMENT
REAL TIME SYSTEMS

Real Time Systems



SOUBI-MSP430 Real Time Kernel

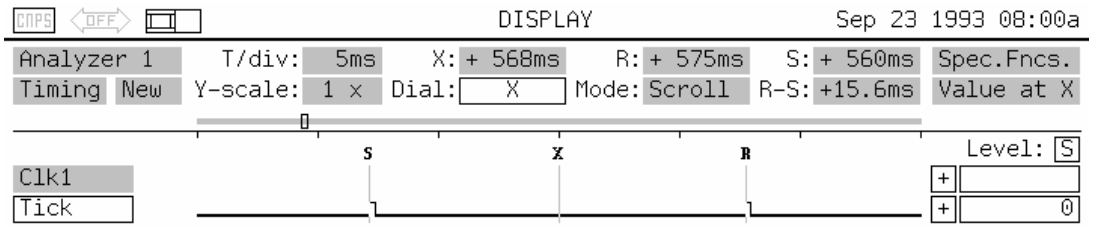
- Kernel specifications:
  - full preemptive;
  - priority scheduling.

- MSP430 resources:
  - WDT → *Tick Source:*
    - mode: interval timer mode;
    - clock source: ACLK;
    - timer interval select: WDT clock source / 512;
  - ACLK: low-frequency 32,768 kHz watch crystals;
  - MCLK = 1,048576 MHz (Default after a PUC);
  - Program memory:
    - 1,2kB Code;
    - 264 B Data;
  - Data memory:
    - 56 + 24*(1 + #Tasks)+ #tasks*32 Byte

SOUBI MSP430 Real Time Kernel

- Kernel overhead (Time resources)
  - tick rate overhead:
    - $Rwc_i^{(n+1)} = C_i + \sum_{k \in hp(i)} \lceil Rwc_i^{(n)}/T_k \rceil \times C_k + \lceil Rwc_i^{(n)}/T_{CLK} \rceil \times C_{CLK} + \sum_{k \in pts} \lceil Rwc_i^{(n)}/T_k \rceil \times C_{PER}$
  - experimental data:

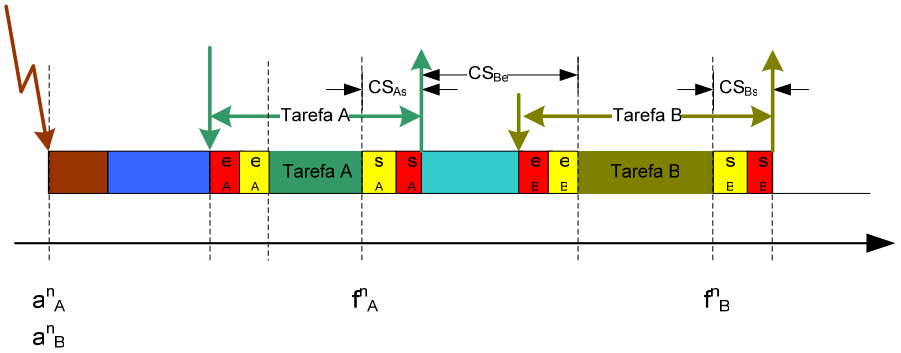
		Activations			
		0	**1**	**2**	**3**
Number of Tasks	**0**	216µs	----------	----------	----------
	1	245µs	249µs	----------	----------
	2	274µs	279µs	313µs	----------
	3	303µs	308µs	343µs	378µs



  - results:
    - T_{CLK} =16 ms
    - C_{CLK} = 216 µs
    - C_{PER} = 5x1st activation + 35x(activations -1 + #Tasks)x 29 µs

SOUBI MSP430 Real Time Kernel

- Context switch overhead:
  - $Rwc_i^{(n+1)} = C_i + CD + \sum_{k \in hp(i)} \lceil Rwc_i^{(n)} / T_k \rceil \times (C_k + CD)$



- experimental data:

Kernel internals	Begin (CD)	End (CD)	TOTAL
_KIntMudTarefa	44,8µs	--	44,8µs
KSheduler	77,3 µs	6,7µs	84,0µs
KTermPeriodica	48,8µs	6,7µs	55,5µs
_KMudTarefa	90µs	--	90µs

- results:
  - CD = 274,3 µs = _KMudTarefa + KSheduler + KTermPeriodica

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

- Objective:
  - shedulability analysis applying the RM algorithm.
- Rate-monotonic scheduling and analysis:
  - processor utilization criterion:
    - Guarantee for shedulability Liu&Layland (1973);
      - $U(n) = \sum^{n}_{i=1}(C_i/T_i) \leq n(2^{1/n}-1)$;
      - $U(n) > 1$ Not schedulable (*overload*);
      - $U(n) \leq n(2^{1/n}-1)$ Schedulable;
      - $1 \geq U(n) \geq n(2^{1/n}-1)$ Uncertain condition;
  - response time analysis:
    - response time in worst case condition:
      - $\forall i,\ Rwc_i = I_i + C_i$;
      - $I_i = \sum_{k \in hp(i)} \lceil Rwc_i/T_k \rceil \times C_k$;
      - $\lceil Rwc_i/T_k \rceil$ is the sum over higher priorities tasks;
    - worst case response time calculations:
      - $Rwc_i^{(n+1)} = C_i + \sum_{k \in hp(i)} \lceil Rwc_i^{(n)}/T_k \rceil \times C_k$.

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

- The problem specifics:
  - technologic Resources:
    - Real Time Kernel SOUBI;
    - IAR (development workbench);
    - Evaluation kit OLIMEX MSP449.
  - MSP430 resources/specifications:
    - MCLK = 1.048576 MHz (Default after a PUC)

Tasks	Ports	Bit	**Function**
Task1	P1	1	Led 1
		5	Task1 monitor
Task2		3	Led 2
		6	Task 2 monitor
Task3		4	Led 3
		7	Task 3 monitor

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

- task model:
  - Cmin = 12,4μs
  - Load(carga) = 1080 x carga μs
  - C = Cmin + load(carga)

```
static void Task_Led1(void)
{
   UBYTE ControlLed = 1;                //function control
   /*Begin → Port P1 initialization*/
    P1OUT_bit.P1OUT_1 = 1;              //led1   = 1;
    P1OUT_bit.P1OUT_5 = 0;              //Task1  = 0;
    P1DIR_bit.P1DIR_1 = 1;
    P1DIR_bit.P1DIR_5 = 1;
   /*End → Port P1 initialization */
    while(1){
   /*Begin → Task body*/
               P1OUT_bit.P1OUT_5 = 1;  //Task1 = 1 -> Execution begin
               if(ControlLed)
               {
                          ControlLed = 0;
                          P1OUT_bit.P1OUT_1 = 0;//led1 = 0 -> led1 = on
               }else{
                          ControlLed = 1;
                          P1OUT_bit.P1OUT_1 = 1;// led1 = 1 -> led1 = off
               }
               load(CARGA_LED1);       // execution time adjustment
               P1OUT_bit.P1OUT_5 = 0;  // Task1 = 1 -> Execution end
               KTermPeriodica();       // Kernel call to task terminate
     /* Begin → Task body */
      }
}
```

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

- Experimental Settings:
  - Settings 1

Task	Execution time [ms]	Activation period [ms]
Task 1	5,325 (**load(5)**)	62,4 (4)
Task 2	54,5 (**load(50)**)	499,2 (32)
Task 3	10,5 (**load(10)**)	249,6 (16)

  - Settings 2

Task	Execution time [ms]	Activation period [ms]
Task 1	54,5(**load(50)**)	124,8 (8)
Task 2	10,5(**load(10)**)	62,4 (4)
Task 3	21,5(**load(20)**)	93,6 (6)

  - Settings 3

Task	Execution time [ms]	Activation period [ms]
Task 1	10,5(**load(10)**)	62,4 (4)
Task 2	27,0(**load(25)**)	93,6 (6)
Task 3	54,5(**load(50)**)	124,8 (8)

  - Settings 4

Task	Execution time [ms]	Activation period [ms]
Task 1	21.5(**load(20)**)	62,4 (4)
Task 3	27,0(**load(25)**)	93,6 (6)
Task 2	54,5(**load(50)**)	124,8 (8)

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

- Results:
  - upper bound Liu&Layland = 0,7798
  - analytical results:

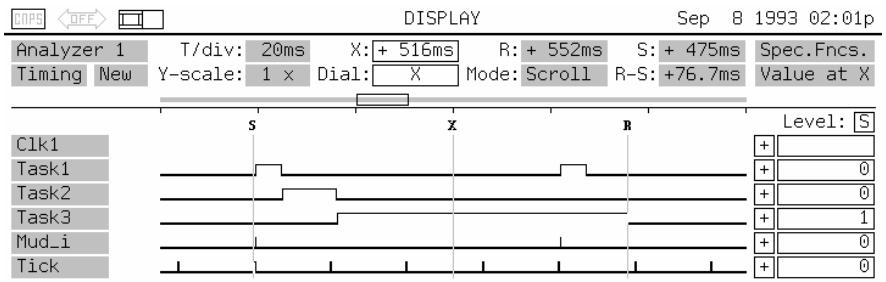
Settings	$T(P_{Low})$ (period)	U	$Rwc(P_{Low})$	condition	shedulability
1	499,2ms	0,24	75,7ms	U < 0,7798	schedulable
2	124,8ms	0,84	118,5ms	$Rwc(P3) < T(P_{Low})$	schedulable
3	124,8ms	0,89	140,0ms	$Rwc(P3) > T(P_{Low})$	not schedulable
4	124,8ms	1,07	--	U > 1	Not schedulable

  - experimental results:

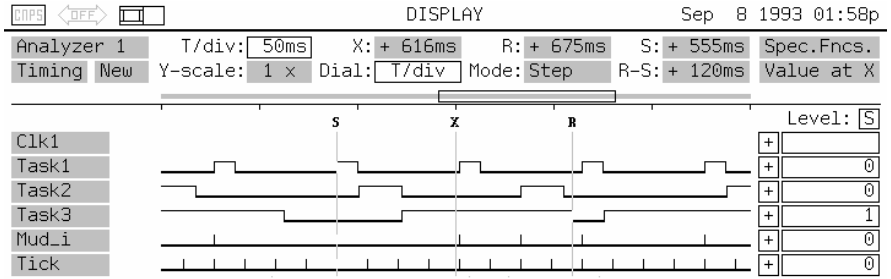
Settings	$Rwc(P_{Low})$	Condition	shedulability
1	76,7ms	$Rwc(P_{Low}) < T(P_{Low}) = 499,2ms$	schedulable
2	120ms	$Rwc(P_{Low}) < T(P_{Low}) = 124,8ms$	schedulable
3	143ms	$Rwc(P_{Low}) > T(P_{Low}) = 124,8ms$	Not schedulable
4	237ms	$Rwc(P_{Low}) > T(P_{Low})$ 124,8ms	Not schedulable

Fixed-Priority Shedulers: Rate – Monotonic algorithm (RM)

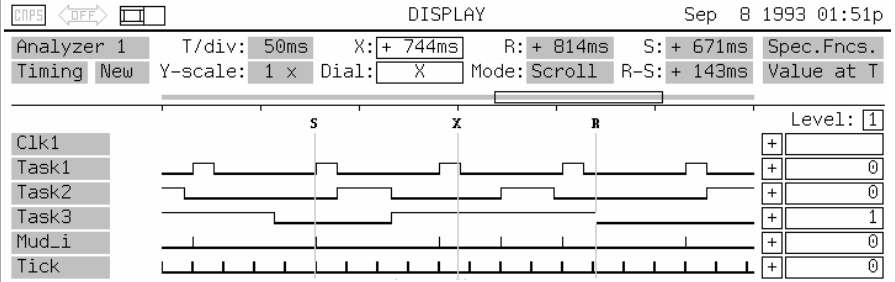
Settings 1 results



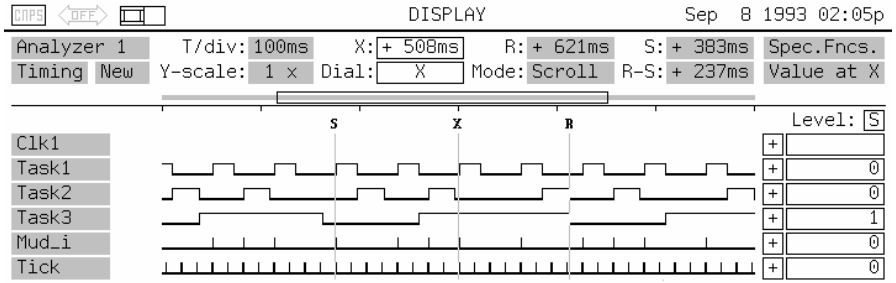
Settings 2 results



Settings 3 results



Settings 4 results



Summary

- Presentation of the teaching structure at UBI in classes that use microprocessors;
- Exposition of the items covered in the microprocessors introduction;
- Examples of the development works included in several Engineering classes;
- Presentation of last year degree projects using MSP430;
- Description of present R&D projects developed by the Microprocessors Working Group @ UBI