**Disciplina:** Sistema Operacionais Embarcados **Código:** 120961 **Turma:** A
**Professor:** Diogo Caetano Garcia
**Aluno/Matrícula:** Fábio Barbosa Pinto – 11/0116356

# Questionário: 15_I2C_2

**1. Considere um MSP430 sendo usado para leituras analógicas. O Raspberry Pi está conectado a ele via I2C, e é o mestre. O MSP430 foi programado para funcionar da seguinte forma:**

- **O MSP430 recebe o byte 0x55, o que indica o começo de conversão.**
- **100us depois, o MSP430 envia o byte menos significativo e o mais significativo da conversão de 10 bits, nesta ordem.**

**Escreva o código para o Raspberry Pi executar este protocolo, de forma a obter conversões a cada 10 ms. A cada 1 segundo ele deve apresentar no terminal a média das últimas 100 amostras.**

```
#include <msp430g2553.h>
#include <legacymsp430.h>

#define LED BIT0
#define BTN BIT3

void init_P1(void);
void init_I2C(void);
void Transmit(unsigned int slave_address, unsigned char data[], unsigned int len);
void Receive (unsigned int slave_address, unsigned char data[], unsigned int len);
void Atraso(volatile unsigned int t);

int main(void)
{
        WDTCTL = WDTPW + WDTHOLD;               // Stop WDT
        BCSCTL1 = CALBC1_1MHZ;
        DCOCTL = CALDCO_1MHZ;
        init_P1();
        init_I2C();
        P1OUT ^= LED;
        Atraso(5000);
        P1OUT ^= LED;
        _BIS_SR(LPM4_bits | GIE);
        return 1;
}

void init_I2C()
{
        UCB0CTL1 |= UCSWRST;                    // Enable SW reset
```

```
        UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC;     // I2C Master, synchronous mode
        UCB0CTL1 = UCSSEL_2 + UCSWRST;            // Use SMCLK, keep SW reset
        UCB0BR0 = 10;                             // fSCL = SMCLK/10 = 100kHz
        UCB0BR1 = 0;
        P1SEL |= BIT6 + BIT7;                     // Assign I2C pins to USCI_B0
        P1SEL2|= BIT6 + BIT7;                     // Assign I2C pins to USCI_B0
        UCB0CTL1 &= ~UCSWRST;                     // Clear SW reset, resume operation
        //IE2 |= UCB0RXIE + UCB0TXIE;             //Enable RX and TX interrupt
}

void Transmit(unsigned int slave_address, unsigned char data[], unsigned int len)
{
        volatile unsigned int i;

        while(UCB0CTL1 & UCTXSTP);            // Ensure stop condition got sent
        UCB0I2CSA = slave_address;
        UCB0CTL1 |= UCTR;
        UCB0CTL1 |= UCTXSTT;            // I2C TX, start condition
        //P1OUT |= LED;
        if(len==1)
        {
                UCB0TXBUF = data[0];
                while(UCB0CTL1 & UCTXSTT);
                //while((IFG2 & UCB0TXIFG)==0);
                UCB0CTL1 |= UCTXSTP;
        }
        else
        {
                UCB0TXBUF = data[0];
                while(UCB0CTL1 & UCTXSTT);
                for(i=1; i<len; i++)
                {
                        UCB0TXBUF = data[i];
                        while((IFG2 & UCB0TXIFG)==0);
                }
                UCB0CTL1 |= UCTXSTP;
        }
        while(UCB0CTL1 & UCTXSTP);
}

void Receive(unsigned int slave_address, unsigned char data[], unsigned int len)
{
        volatile unsigned int i;
        UCB0I2CSA = slave_address;
        while(UCB0CTL1 & UCTXSTP);            // Ensure stop condition got sent
        UCB0CTL1 &= ~UCTR ;                   // Clear UCTR
        UCB0CTL1 |= UCTXSTT;                  // I2C start condition
```

```
        while(UCB0CTL1 & UCTXSTT);              // Start condition sent?

        for(i=0; i<len; i++)
        {
                if((len-i)==1)
                        UCB0CTL1 |= UCTXSTP;
                while((IFG2 & UCB0RXIFG)==0);
                data[i] = UCB0RXBUF;
        }
        while(UCB0CTL1 & UCTXSTP);
}

void init_P1(void)
{
        P1OUT &= ~LED;
        P1DIR |= LED;
        P1DIR &= ~BTN;
        P1REN |= BTN;
        P1OUT |= BTN;
        P1IES |= BTN;
        P1IE  |= BTN;
}

// Atraso de t*100 us
void Atraso(volatile unsigned int t)
{

        TACCR0 = 100-1;
        TACTL |= TACLR;
        TACTL = TASSEL_2 + ID_0 + MC_1;
        while(t--)
        {
                while((TACTL&TAIFG)==0);
                TACTL &= ~TAIFG;
        }
        TACTL = MC_0;
}

interrupt(PORT1_VECTOR) P1_ISR(void)
{
        volatile unsigned int MSP430_slave = 0xAD;
        unsigned char t = 0x55, rcv[2];
        unsigned int d=0;
        while((P1IN & BTN)==0);
        Transmit(MSP430_slave, &t, 1);
        Atraso(1);
        Receive(MSP430_slave, rcv, 2);
```

```
        d = (unsigned int)rcv[1];
        d = (d<<8) | ((unsigned int)rcv[0]);
        //P1OUT &= ~LED;
        if(d>1020)
        {
                Atraso(5000);
                P1OUT |= LED;
                Atraso(5000);
                P1OUT &= ~LED;
        }
        P1IFG = 0;
}
```