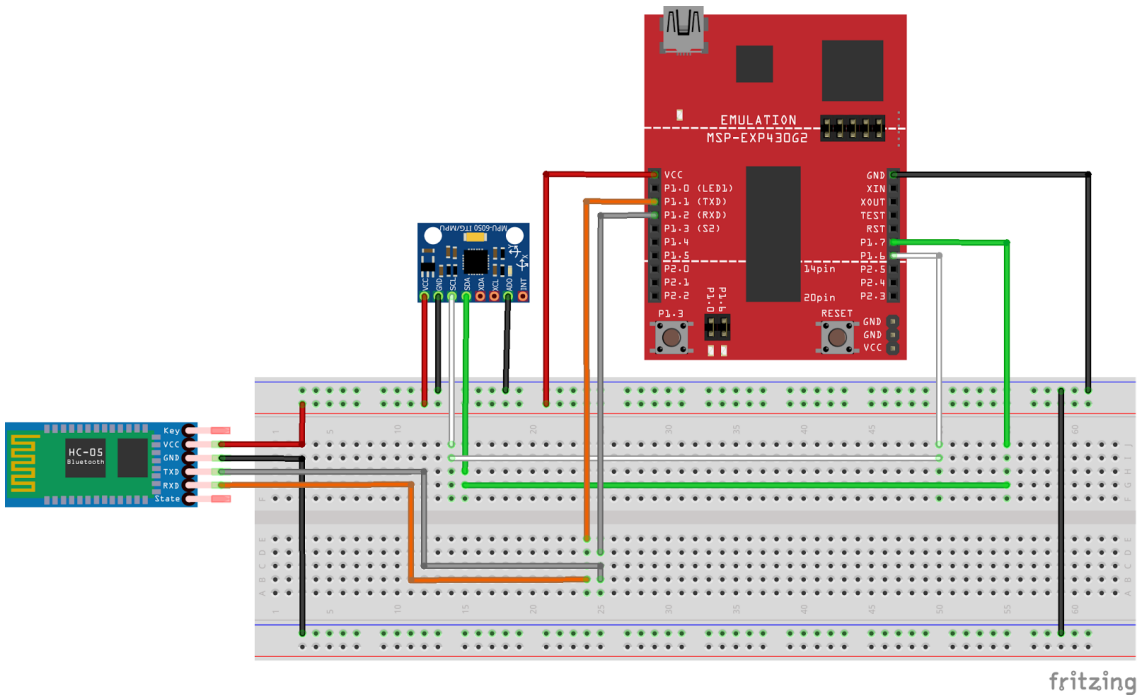
Bateria virtual controlada pela Launchpad MSP430G2553

Filipe de Souza Freitas
FGA
Universidade de Brasília
Gama, Brasil
filipe.desouzafreitas@gmail.com

Rodrigo Magalhães Caires
FGA
Universidade de Brasília
Gama, Brasil
rodrigo_macrt@hotmail.com

Palavras-chave—Microcontrolador, MSP430, Percussão, Bluetooth, I2C, UART.

Fig. 1. Esquema de pinagem.



A figura 1 demonstra o esquema a ser montado para estabelecer comunicação com dispositivos externos e fazer leitura do sensor a fim de executar um arquivo áudio em momento apropriado.

Comunicação I2C:

A comunicação I2C é composta por dois sinais de comunicação : SDA e SCL.

- SDA, onde são transmitidos os dados entre os dispositivos;
- SCL é o clock para a comunicação.

Uma comunicação I2C entre o dispositivo master(responsável pelo controle da comunicação,

gera o clock) e o dispositivo slave(dispositivo que recebe “ordens” do master) tem como característica:

1. Condição de START, quando a linha SDA está em borda de descida e SCL está em nível alto.
2. O dispositivo master envia através do bus SDA o endereço do dispositivo slave.
3. O slave responde gerando uma condição de ACK(Acknowledge) .
4. Condição de Stop, quando a linha SDA está em borda de subida(transição de nível lógico baixo para nível lógico alto) e SCL está em nível alto.

Existem várias formas de comunicação entre dispositivos I2C, entre elas, podemos citar:

- Master Receiver(master recebe dados o slave)
- Master Send(Master envia dados ao slave)
- Slave Receiver(Slave recebe dados do master)
- Slave Send(slave manda dados ao master)
- Multimaster (mais de um master com um deles como receptor e outro como transmissor)

I2C com dispositivo MPU6050:

Através de dados obtidos no datasheet o para um sistema single master receiver/transmitter os protocolos de comunicação são descritos a seguir:

Operação de escrita de dados (Escrever dados nos registradores do Mpu6050) para mandar um único byte:

1. Condição de START.
2. AD+W -> Master envia : Slave address + write bit.
3. Slave envia ACK.
4. RA -> Master envia : Register Address to be write.
5. Slave envia ACK.
6. Dados para RA.
7. Slave envia ACK.
8. Condição de STOP.

A operação começa com o dispositivo master gerando a condição de START e escrevendo o endereço do dispositivo slave no SDA, logo após receber uma condição de ACK o dispositivo master informa o endereço que deseja escrever os dados e espera por outra condição de ACK a ser enviada pelo slave, logo após master envia os dados a serem escritos, quando o slave gerar a condição de ACK o master gera a condição de STOP e a comunicação termina. Caso seja necessário enviar mais de um byte ao invés de

Operação de leitura de um byte de dados:

1. Condição START.
2. AD+W -> Master evia : Slave address + write bit.
3. Slave envia ACK
4. RA -> Master envia: Endereço a ser lido.
5. Slave envia ACK
6. Condição START .
7. AD+R -> Master envia : Slave address + read bit
8. Slave envia ACK
9. Slave envia Data
10. Master envia NACK e condição de Stop.

A operação começa com o dispositivo master gerando a condição de START e escrevendo o endereço do dispositivo slave no SDA, logo após receber uma condição de ACK o dispositivo master informa o endereço que deseja ler, quando o slave gera uma condição de ACK seguido dos dados no endereço solicitado pelo master, o dispositivo master ao receber os dados gera condição de NACK e de STOP terminando assim a comunicação.

Para receber mais de um byte pode-se começar e encerrar a comunicação mudando-se o endereço do registrador do dispositivo slave a ser lido pelo dispositivo master

Comunicação I2C UART:
No modo UART o msp430g2553 contém uma estrutura de hardware que lida com o interfaceamento I2C e um conjunto de registradores para utilizar a comunicação. São eles:

Table 1. USCI_B0 Registradores de Controle.

Registrador	Forma curta(Nome)	Tipo	Endereço	Estado inicial
USCI_B0 control register 0	UCB0CTL0	Read/write	068h	001h with PUC
USCI_B0 control register 1	UCB0CTL1	Read/write	069h	001h with PUC
USCI_B0 bit rate control register 0	UCB0BR0	Read/write	06Ah	Reset with PUC
USCI_B0 bit rate control register 1	UCB0BR1	Read/write	06Bh	Reset with PUC
USCI_B0 I2C interrupt enable register	UCB0I2CIE	Read/write	06Ch	Reset with PUC
USCI_B0 status register	UCB0STAT	Read/write	06Dh	Reset with PUC
USCI_B0 receive buffer register	UCB0RXBUF	Read	06Eh	Reset with PUC
USCI_B0 transmit buffer register	UCB0TXBUF	Read/write	06Fh	Reset with PUC
USCI_B0 I2C own address register	UCB0I2COA	Read/write	0118h	Reset with PUC
USCI_B0 I2C slave address register	UCB0I2CSA	Read/write	011Ah	Reset with PUC
SFR interrupt enable register 2	IE2	Read/write	001h	Reset with PUC
SFR interrupt flag register 2	IFG2	Read/write	003h	00Ah with PUC

Em geral os registradores acima citados constituem as seguintes funções de hardware:

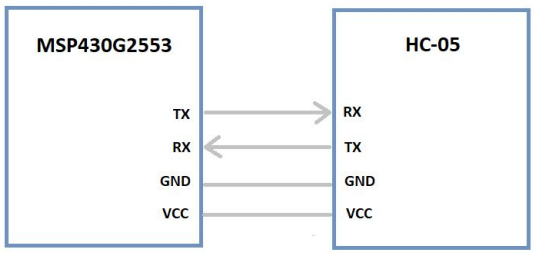
- UCB0_CTL0 Modo de operação(SPI,I2C,UART e etc ...).
- UCB0_CTL1 Controle do fluxo de operação (start, stop e etc).
- UCBR0(low byte) + UBR1(high byte) Controla a frequência de operação de SCL.
- UCB0Sat Registrador de status.
- UCB0RxBUF buffer para receber dados através do pino SDA(pino P1.7).
- UCB0TxBUF buffer para transmitir dados através do pino SDA(pino P1.7).

- UCB0I2COA Endereço do prÃ³rpio msp430 na comunicação I2C.
- UCB0I2CSA escravo na comunicação I2C.
- UCB0_I2CIE Registrador de habilitação de interrupção para I2C.
- IE2 Habilitar interrupção.
- IFG Flag de interrupção.

Comunicação bluetooth

Para executar a comunicação remota entre a Launchpad e o smartphone ou computador, utilizamos um módulo bluetooth HC-05. A metodologia de comunicação utilizada foi a UART (Universal Asynchronous Receiver Transmitter), capaz de transmitir ou receber informações de até 8 por vez a partir de dois canais seriais, Rx (receptor) e Tx (transmissor).

Figura 2 - Diagrama de blocos.



Como não há o compartilhamento de um sinal de clock, ambos os dispositivos devem estar em acordo quanto à frequência de troca de informações. Para isso, tanto o dispositivo mestre quanto o escravo devem ser configurados sob a mesma baud rate (taxa de variação de sinais por segundo). Nesse caso, o módulo HC-05 já vem com a baud rate pré-definida de 9600 bits/s então basta configurar a Launchpad com a mesma taxa. Utilizando o datasheet da MSP430G2553 e definindo o clock da placa em 1MHz, podemos ver que o valor tabelado para uma baud rate de 9600 nessas circunstâncias é configurado quando o registrador UCBR0 recebe 104.

Como os pinos da MSP430G2553 são multiplexados, eles podem exercer funções diferentes dependendo da sua configuração. No caso do protocolo UART, os pinos Rx e Tx se encontram nos pinos P1.1 e P1.2 respectivamente. Porém, por default, estes pinos são inicialmente definidos como GPIO, ou seja, como pinos de entrada ou saída de propósito geral. Para trocar as funcionalidades destes e outros pinos, utilizamos os registradores de 8 bits P1SEL e P1SEL2. Ao setar os bits 1 e 2 (referentes aos pinos P1.1 e P1.2, respectivamente) em ambos os registradores, habilitamos as funcionalidades RXD e TXD para a UART.

Em seguida, “setam” se os interrupt enables do receptor, do transmissor e o global. A flag do transmissor é setada automaticamente quando um dado é enviado, ou seja, quando o registrador UCA0TXBUF está vazio, e é “resetada” automaticamente quando um novo dado é escrito no mesmo registrador. Já a flag do receptor é setada automaticamente quando um dado é recebido e armazenado no registrador UCA0RXBUF e precisa ser “resetada” manualmente sempre que for receber um novo dado.

Para o exemplo do código anexado (Anexo I), o registrador UCA0TXBUF (transmitter) recebe o caracter ‘A’. A Launchpad pode ser ainda colocada em um estado de loop back mode para testes, onde a saída Tx envia o dado diretamente para o Rx da própria MSP por meio de uma conexão interna e serve para verificar se a comunicação está sendo realizada corretamente. O loop back mode pode ser configurado setando o bit referente ao UCLISTEN do registrador de status UCA0STAT.

Por fim, o teste com o bluetooth foi realizado através do programa PuTTY. Ao carregar o código do Anexo I na Launchpad utilizando o IAR Workbench e a montagem seguiu, a grosso modo, o modelo do diagrama de blocos da figura 1, conectando se o pino P1.1 da MSP430 ao TXD do HC-05, o P1.2 ao RXD e os pinos VCC e GND aos seus relativos. Com o módulo bluetooth ligado, ativamos o bluetooth do notebook para o emparelhamento. Em seguida, identificamos as portas COM que o módulo utilizava para se conectar ao computador através da aba “Gerenciador de Dispositivos” do Windows. Em seguida, abrimos o PuTTY e escolhemos a opção “Serial” na seção “Connection type” e digitamos a porta identificada no passo anterior no espaço “Serial line”. Finalmente clicamos em Open e observamos o dado, neste caso o caracter ‘A’, chegar ao terminal do PuTTY toda vez que apertamos o botão de reset da Launchpad.

Referência

[1] User guide
[2] Data sheet msp430g2553
[3]https://www.youtube.com/watch?v=6IAkYpmA1DQ(tutorial arduino)
[4]https://www.youtube.com/watch?v=KPZRoBmjjFM&t=136s(How to Set Up the Circuit for I2C or TWI)
[5]https://www.youtube.com/watch?v=7aDjYqOMu9w

How to Initialize and Communicate with I2C or TWI (1 of 2) - Tutorial on I2C TWI Part 3

[6]https://www.youtube.com/watch?v=MFrQWTVfX3k&t=3s
(How to Code and Understand the I2C/TWI (Two Wire Interface) - A Tutorial - Part 1)
[7]Data sheet mpu6050
[8]i2cdev-master(github)

Anexo I -

```
#include "msp430g2553.h"

void main(void)

{

  WDTCTL = WDTPW + WDTHOLD; // Stop the Watch dog


     //-------------------- Configure the Clocks -------------------//


  if (CALBC1_1MHZ==0xFF)  // If calibration constant erased

    {

      while(1);        // do not load, trap CPU!!

    }

   DCOCTL  = 0;           // Select lowest DCOx and MODx settings

  BCSCTL1 = CALBC1_1MHZ;  // Set range

   DCOCTL  = CALDCO_1MHZ;  // Set DCO step + modulation


     //----------------- Configuring the LED's ----------------------//


    P1DIR  |=  BIT0 + BIT6;  // P1.0 and P1.6 output

   P1OUT  &= ~BIT0 + BIT6;  // P1.0 and P1.6 = 0


  //--------- Setting the UART function for P1.1 & P1.2 --------//


    P1SEL  |= BIT1 + BIT2;  // P1.1 UCA0RXD input

   P1SEL2 |= BIT1 + BIT2;  // P1.2 UCA0TXD output


  //------------ Configuring the UART(USCI_A0) ----------------//


   UCA0CTL1 |= UCSSEL_2 + UCSWRST;  // USCI Clock = SMCLK,USCI_A0 disabled

   UCA0BR0   =  104;              // 104 From datasheet table-

    UCA0BR1    =  0;                  // -selects baudrate =9600,clk = SMCLK

     UCA0MCTL   =  UCBRS_1;             // Modulation value = 1 from datasheet

   //UCA0STAT |= UCLISTEN;          // loop back mode enabled

  UCA0CTL1 &= ~UCSWRST;          // Clear UCSWRST to enable USCI_A0


     //----------------- Enabling the interrupts ------------------//


   IE2 |= UCA0TXIE;               // Enable the Transmit interrupt

   IE2 |= UCA0RXIE;               // Enable the Receive  interrupt
```

```
   _BIS_SR(GIE);                     // Enable the
global interrupt


   UCA0TXBUF = 'A';                  // Transmit a
byte


   _BIS_SR(LPM0_bits + GIE);         // Going to
LPM0

}


//-------------------------------------------------------
---------------//

  //                Transmit and Receive interrupts
//


//-------------------------------------------------------
---------------//

 #pragma vector = USCIAB0TX_VECTOR

 __interrupt void TransmitInterrupt(void)

 {

   P1OUT  ^= BIT0;//light up P1.0 Led on Tx

 }

 #pragma vector = USCIAB0RX_VECTOR

 __interrupt void ReceiveInterrupt(void)

 {

    P1OUT   ^= BIT6;     // light up P1.6 LED on
RX

   IFG2 &= ~UCA0RXIFG; // Clear RX flag

 }
```