Projeto Final de Arquitetura de Computadores

Uso de Microcontroladores - Relatório.

|  |  |
| --- | --- |
| Departamento de Computação – UFRN – Ciência e Tecnologia 2016.1 | |
| Componentes | Lukas Maximo Grilo Abreu Jardim |
| Maxon Ramon dos Anjos Medeiros |
| Damiano Alves de Lima |
| Professores | Diogo Pinheiro Fernandes Pedrosa |
| Sérgio Natan Silva |

# Relatório

Nesse trabalho, vamos programar um micro controlador da Atmel, o ATmega328P, através de uma placa Arduino UNO, para simular o funcionamento de um sensor de presença acionado através de um LDR, um Resistor de Foto-sensibilidade, para acionar e desativar a interrupção quando necessário.

Para isso, inicialmente, fizemos um código em C no AVR Studio, mas não foi possível executar o código, por não obter conexão bem sucedida da entre o código na IDE “AVR Studio”, por falta de uma biblioteca. Mas havia um plano B: a IDE do Arduino, onde foi feito o código para executar os passos do projeto, acender o LED, a conversão Analógico/Digital, a captação das leituras do potenciômetro, e o sinal de interrupção e o mouse funcionaram como esperado.

Contudo o trabalho tem evoluído bem, devido ás ideias dos membros do grupo que ajudaram bastante e ajudaram a realizar o projeto.

# Código

#include "PAN3101.h"

// #include "ADNS2051.h"

// #include "ADNS2610.h"

// #include "ADNS2083.h"

#define SCLK 4 // Serial clock pin on the Arduino

#define SDIO 5 // Serial data (I/O) pin on the Arduino

PAN3101 Optical1 = PAN3101(SCLK, SDIO);

// ADNS2051 Optical1 = ADNS2051(SCLK, SDIO);

// ADNS2610 Optical1 = ADNS2610(SCLK, SDIO);

// ADNS2083 Optical1 = ADNS2083(SCLK, SDIO);

signed long x2 = 0;

signed long y2 = 0;

int potPin = A0;

int pinA = 7;

int ledp3 = 10;

int ledPin = 13;

int ledp2 = 8;

volatile unsigned int val = 0;

int v2 = 500;

byte b = 0;

byte c = 0;

byte d = 0;

int interrupt = 2;

byte valor = 0;

volatile int interruption = 0;

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(pinA, OUTPUT);

pinMode(ledp3, OUTPUT);

pinMode(ledp2, OUTPUT);

pinMode(interrupt, OUTPUT);

Serial.begin(9600);

}

void Optical(){

//delay(100);

x2 += Optical1.dx();

y2 += Optical1.dy();

Serial.print("x=");

Serial.print(x2, DEC);

Serial.print(" y=");

Serial.println(y2, DEC);

}

void functionInterruption(){

Optical();

delay(1000);

Serial.print("Number of Interruption is ");Serial.println(interruption);

Serial.print("x=");

Serial.print(x2, DEC);

Serial.print(" y=");

Serial.println(y2, DEC);

digitalWrite(ledp3,HIGH);

delay(1000);

digitalWrite(ledp3,LOW);

delay(1000);

interruption++;

val = analogRead(potPin);

map(val,0,65535,0,255);

valor = val;

if(valor < 100){

detachInterrupt(0);

loop();

}

}

void funciona(){

digitalWrite(pinA,HIGH);

delay(100);

val = analogRead(potPin);

map(val,0,65535,0,255);

valor = val;

v2 = analogRead(potPin);

b = val;

digitalWrite(ledPin, HIGH);

//c = v2;

//d = b + c;

Serial.print("A = ");

Serial.print(valor);

Serial.print(";B = ");

Serial.print(b);

Serial.println(";");

if(valor > 100){

digitalWrite(pinA, LOW);

digitalWrite(ledPin,LOW);

attachInterrupt(0, functionInterruption, LOW);

digitalWrite(ledp2, HIGH);

}else{

digitalWrite(pinA, HIGH);

digitalWrite(ledp2, LOW);

}

}

void loop() {

funciona();

Optical();

}

# Análise do Código

Foram necessárias duas funções (além das funções ‘setup’ e ‘loop’), para executar o código e uma para a interrupção: a função ‘setup’ inicializa os pinos 13, 10, 08 e 07 do Arduino, a função ‘loop’ executa o mouse pela função ‘Optical’ e o LDR pela função ‘funciona’. Além disso, na segunda função, quando o valor do potenciômetro (LDR) é maior que um determinado sinal crítico, A função de interrupção é acionada, colocando o pino 07 em estado baixo, e ela só vai ser desabilitada quando o valor do potenciômetro não satisfazer a condição de interrupção.

Os pinos 13, 10 e 8, executa o funcionamento das LEDs, o pino A0 é acionado no início da execução da função ‘funciona’, o pino 07 é conectado ao pino 02, responsável pelo sinal de Interrupção, e os pinos 04 e 05 são conectados ao mouse, o pino 04 transmite o sinal de relógio (CLOCK) e o pino 05 transmite os dados do mouse para a placa Arduino.

Para que isso funcionasse corretamente, foi necessário adicionar resistores para proteger os LEDs e o potenciômetro (LDR).

# Conclusão

Para realizar esse trabalho foi necessário analisar as entradas do microcontrolador ATmega328P, para encontrar as saídas da placa Arduino, coincidentes com as do microcontrolador da Atmel.