Matthew Russell, Joshua Ramayrat

Professor Wolfe

ELEN123, Mechatronics

Laboratory #3: Interrupts and Timers

**Experiment 1:**

It was asked to write a program that would trigger an interrupt service routine and subsequently print a serial output count of the amount of times a switch button has been pressed.

*volatile int count, sw;*

*void setup() {*

*Serial.begin(9600);*

*pinMode(2, INPUT\_PULLUP);*

*attachInterrupt(0, isrf, FALLING);*

*count = 0;*

*sw = 0;*

*}*

*void loop() {*

*if (sw==1){*

*Serial.println(count);*

*sw = 0;*

*}*

*}*

*void isrf() {*

*count++;*

*sw = 1;*

*}*

There were multiple counts of the switch button which threw off the data. This was due to the vibrations of the switch - multiple vibrations will give several on and off interactions from pin 2 to ground, giving multiple counts in the serial console.

**Experiment 2:**

It was asked to rewire the original circuit to account for the noise and vibrations of the switch-button press. An inverter 74HC14 was used connected to the input pin 2 of the arduino. Because it was an inverter, the interrupt must be called on the rising edge of the signal.

*volatile int count, sw;*

*void setup() {*

*Serial.begin(9600); //initialize serial console*

*pinMode(2, INPUT\_PULLUP); //initialize input pin 2 with the built-in pullup resistor.*

*attachInterrupt(0, isrf, RISING); //attach an interrupt service routine to a rising signal on pin 2.*

*count = 0;*

*sw = 0;*

*}*

*void loop() {*

*if (sw==1){*

*Serial.println(count);*

*sw = 0;*

*}*

*}*

*void isrf() {*

*count++;*

*sw = 1;*

*}*

**Experiment 3:**

It was asked to write a program to generate a TimerOne interrupt every 100 ms; add the amount of times it’s called to a count variable; and count the amount of elapsed time every 1 second. This program is as follows:

*#include <TimerOne.h>*

*// This example uses the timer interrupt to blink an LED*

*// and also demonstrates how to share a variable between*

*// the interrupt and the main program.*

*const int led = LED\_BUILTIN; // the pin with a LED*

*void setup(void)*

*{*

*pinMode(led, OUTPUT);*

*Timer1.initialize(100000);*

*Timer1.attachInterrupt(blinkLED); // blinkLED to run every 0.15 seconds*

*Serial.begin(9600);*

*}*

*// The interrupt will blink the LED, and keep*

*// track of how many times it has blinked.*

*int ledState = LOW;*

*volatile unsigned long blinkCount = 0, t = 0; // use volatile for shared variables*

*volatile int inc = 0;*

*void blinkLED(void)*

*{*

*if (ledState == LOW) {*

*ledState = HIGH;*

*blinkCount = blinkCount + 1; // increase when LED turns on*

*} else {*

*ledState = LOW;*

*}*

*digitalWrite(led, ledState);*

*inc++;*

*}*

*// The main program will print the blink count*

*// to the Arduino Serial Monitor*

*void loop(void)*

*{*

*unsigned long blinkCopy; // holds a copy of the blinkCount*

*// to read a variable which the interrupt code writes, we*

*// must temporarily disable interrupts, to be sure it will*

*// not change while we are reading. To minimize the time*

*// with interrupts off, just quickly make a copy, and then*

*// use the copy while allowing the interrupt to keep working.*

*noInterrupts();*

*blinkCopy = blinkCount;*

*interrupts();*

*if(inc == 10){*

*t = 200\*blinkCopy;*

*Serial.print("elapsed time (sec) is: ");*

*Serial.println(t/1000);*

*inc = 0;*

*}*

*}*

**Experiment 4:**

It was asked to write a program to generate an interrupt on the rising edge of channel A and check the value on channel B. A counter variable should be incremented in one direction and decremented in another direction. This program is as follows:

*//Matt Russell, Joshua Ramayrat*

*//Lab 3 Exercise 4.3*

*//4/24/17*

*//Pin 2 is channel A, Pin 3 is channel B*

*volatile int count = 0, sw = 0;*

*void setup()*

*{*

*pinMode(2,INPUT\_PULLUP);*

*pinMode(3,INPUT\_PULLUP);*

*attachInterrupt(0, isr0, RISING);*

*Serial.begin(9600);*

*}*

*//Value of B at rising edge of A tells direction of motion*

*//right->B = HIGH, left->B = LOW*

*void isr0(void)*

*{*

*if(digitalRead(3)){*

*count++;*

*}*

*else{*

*count--;*

*}*

*sw = 1;*

*}*

*void loop() {*

*if(sw == 1){*

*Serial.println(count);*

*sw = 0;*

*}*

*}*

It was also asked to modify the program to measure the distance travelled by the wheel in both directions:

*//Matt Russell, Joshua Ramayrat*

*//Lab 3 Exercise 4.4*

*//4/25/17*

*//Pin 2 is channel A, Pin 3 is channel B*

*//Computes distance traveled by wheel*

*volatile int count = 0, sw = 0;*

*void setup()*

*{*

*pinMode(2,INPUT\_PULLUP);*

*pinMode(3,INPUT\_PULLUP);*

*attachInterrupt(0,isr0,RISING);*

*Serial.begin(9600);*

*}*

*//want distance traveled so just increment count on each interrupt*

*void isr0(void){*

*count++;*

*sw = 1;*

*}*

*void loop()*

*{*

*if(sw == 1){*

*Serial.print(count\*(PI/180)\*10); //printout distance traveled*

*Serial.println("meters traveled");*

*sw = 0;*

*}*

*}*

**Experiment 5:**

It was asked to write a program to blink an LED for 200 ms every 500 ms; count the meters travelled by the wheel; and increment a count to compute how many of these counts correspond to one full wheel rotation. The program is as follows:

*#include <TimerOne.h>*

*const int led = 13; // the pin with a LED.*

*long interval = 200; //blink LED for 200 ms.*

*void setup(void)*

*{*

*pinMode(led, OUTPUT);*

*pinMode(2, INPUT\_PULLUP);*

*pinMode(3, INPUT\_PULLUP);*

*attachInterrupt(0, isr0, RISING);*

*attachInterrupt(0, isr0, FALLING);*

*Timer1.initialize(500000);*

*Timer1.attachInterrupt(blinkLED); // blinkLED to run every 500 ms.*

*Serial.begin(9600);*

*}*

*int ledState = LOW;*

*long previousMillis = 0;*

*unsigned long currentMillis = millis();*

*void blinkLED(void) //LED interrupt*

*{*

*unsigned long currentMillis = millis();*

*if(currentMillis - previousMillis > interval) {*

*previousMillis = currentMillis;*

*if (ledState == LOW)*

*ledState = HIGH;*

*else*

*ledState = LOW;*

*digitalWrite(led, ledState);*

*}*

*}*

*volatile int count = 0, sw = 0;*

*void isr0(void)*

*{*

*count++;*

*sw=1;*

*}*

*void loop(void)*

*{*

*blinkLED();*

*if (sw ==1){*

*Serial.print(count\*(PI/180)\*10);*

*Serial.println(" meters traveled");*

*Serial.print("Count: ");*

*Serial.println(count);*

*Serial.println(" ");*

*sw = 0;*

*}*

*}*

**Experiment 6:**

It was asked to report the distance to the wall using a timer interrupt every 250 ms and copy the serial monitor output.

*#include <TimerOne.h>*

*int pinButton = 30;*

*float distance;*

*void setup() {*

*Serial.begin(9600);*

*pinMode(A0, INPUT); //IR sensor output connects to analog input.*

*pinMode(pinButton, INPUT\_PULLUP); //pushbutton switch connects to digital input.*

*Timer1.initialize(250000);*

*Timer1.attachInterrupt(distanceWall);*

*}*

*void loop() {}*

*void distanceWall(){*

*float sensorValue = analogRead(A0);*

*float voltage = sensorValue \* (5.0 / 1023.0);*

*Serial.println(voltage);*

*if (voltage <= 2 && voltage >= 1) {*

*distance = (-5.9242\*voltage) + 16.434;*

*} else if (voltage < 1 && voltage >= 0.75) {*

*distance = (-16.129\*voltage) + 27.516;*

*} else if (voltage < 0.75 && voltage >= 0.51) {*

*distance = (-39.447\*voltage + 45.299);*

*} else if (voltage < 0.51 && voltage > 0.3) {*

*distance = (-122.24\*voltage) + 90.107;*

*}*

*Serial.println(distance);*

*}*

It was asked to modify the existing code to have a timer interrupt every 25 ms, average 10 measurements, and print out the average every 250 ms:

*#include <TimerOne.h>*

*int pinButton = 30;*

*float distance;*

*float totalDistance = 0;;*

*int avgCount = 0;*

*float average;*

*void setup() {*

*Serial.begin(9600);*

*pinMode(A0, INPUT); //IR sensor output connects to analog input.*

*pinMode(pinButton, INPUT\_PULLUP); //pushbutton switch connects to digital input.*

*Timer1.initialize(2500000);*

*Timer1.attachInterrupt(distanceWall);*

*}*

*void loop() {*

*if (avgCount = 10){*

*average = totalDistance/10;*

*Serial.println("The average distance over a period of 250 ms is: ");*

*Serial.println(average);*

*}*

*}*

*void distanceWall(){*

*float sensorValue = analogRead(A0);*

*float voltage = sensorValue \* (5.0 / 1023.0);*

*Serial.println(voltage);*

*if (voltage <= 2 && voltage >= 1) {*

*distance = (-5.9242\*voltage) + 16.434;*

*} else if (voltage < 1 && voltage >= 0.75) {*

*distance = (-16.129\*voltage) + 27.516;*

*} else if (voltage < 0.75 && voltage >= 0.51) {*

*distance = (-39.447\*voltage + 45.299);*

*} else if (voltage < 0.51 && voltage > 0.3) {*

*distance = (-122.24\*voltage) + 90.107;*

*}*

*int avgCount = avgCount + 1;*

*totalDistance = totalDistance + distance;*

*}*