CG1112: Engineering Principles and Practices II

Week 7: Studio 2: Arduino ADC Programming (GRADED LAB)

Objectives:

1. Understand how the Atmega328's ADC peripheral block needs to be configured.
2. Develop Low-Level (Bare-Metal) code for the ADC module for both the Polling and Interrupt Modes of Operation.

Equipment Needed:

1. Laptop with Arduino IDE installed
2. Arduino Uno Board + Prototyping Board
3. 10k Pot + 330ohms Resistor
4. Red LED (x1)

LAB REPORT:

- Remember to show any necessary workings.
- Waveform images can be captured through your camera or they can be redrawn in your report.
- Use the Template given in IVLE for your submission.

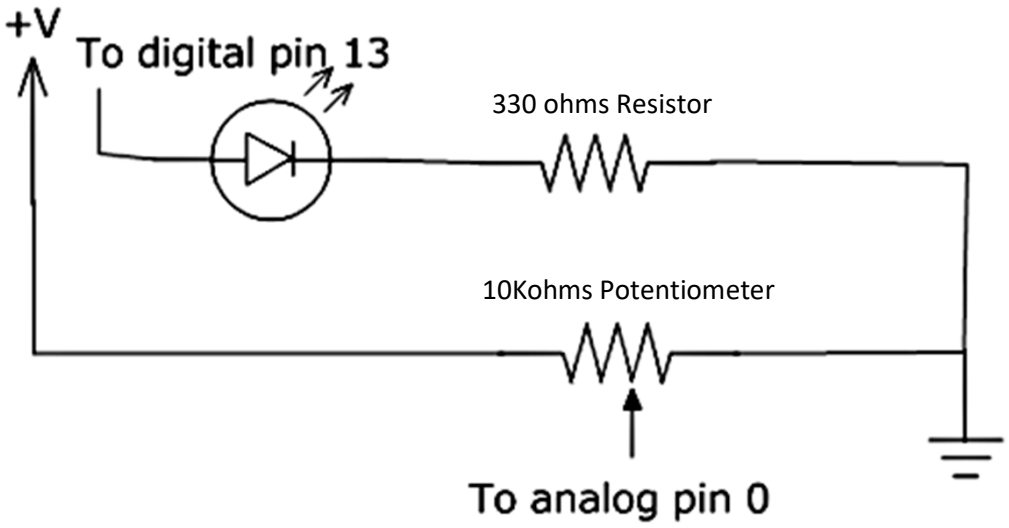
1. Introduction

In EPP1, you dealt with Analog Devices in the MBot in the form of IR sensors in order to detect the right and left walls. If you recall, you used the built-in adc library calls to get the sensor value. In this studio, you are going to be using the same ADC modules, but you are going to learn how to program the module bare-metal.

In the e-lecture, you should have gotten an idea on how the ADC module in the microcontroller helps us to achieve our objective. By the end of this studio, you should be able to configure the ADC module and use it to perform the necessary conversions.

2. Exercise A

2.1 A sample code for the ADC module is provided in Appendix_A. Compile and download it to your Uno board. Connect your circuit as shown below.



Mapping:

Analog Pin 0 -> Port C Pin 0 (A0 on the Uno)

Digital Pin 13 -> Port B Pin 5

Observe the behavior of the LED as you tune the potentiometer from one end to the other.

LAB REPORT

Q1. Tune the potentiometer to the LOWEST possible value and measure its resistance. What is the equivalent voltage reading when the potentiometer is at this setting?

Lowest Possible R Value:

Voltage Reading:

Q2. Tune the potentiometer to the HIGHEST possible value and measure its resistance. What is the equivalent voltage reading when the potentiometer is at this setting?

Highest Possible R Value:

Voltage Reading:

Q3. With the potentiometer set to the LOWEST value, observe the waveform at Pin 13 of the Uno. Measure the Mean Value and the Frequency of the signal. Include a copy of the waveform in your report.

Mean Value:

Frequency:

Q4. With the potentiometer set to the LOWEST value, observe the waveform at Pin 13 of the Uno. Measure the Mean Value and the Frequency of the signal. Include a copy of the waveform in your report.

Mean Value:

Frequency:

Q5. Explain why we don't see any change in the LED Brightness with the current code?

3. Exercise B

In the first exercise, you implemented the ADC operation using the Polling approach. We will now implement it using Interrupts. Examine the code given in Appendix_B. Complete the code in the ISR, compile and download the code onto the board.

LAB REPORT

Q6. Provide the code for the ISR.

```
ISR(ADC_vect)
{


}
```

LAB REPORT

Q7. Tune the potentiometer to achieve the HIGHEST possible frequency of the square was at Pin 13. Take note of its frequency and explain any differences with the value obtained in Q3.

4. Exercise C

With the Duty Cycle still set at 50%, we still don't see any change in the LED brightness. Now let's incorporate the PWM functionality from Studio 6 so that we can change the brightness of the LED as we tune the potentiometer. The ADC value must be used to change the PWM Duty-cycle so that we can observe a change in the brightness of the LED.

LAB REPORT

Q8. Provide the full source code for your solution to Exercise C.

You must ensure that it goes through only one cycle of fading as we tune the potentiometer from one end to the other.

Summary

In this studio, you have learnt how to program the ADC module of the AT328P using the low-level registers. The ADC module is integral in dealing with many types of sensors and if you intend to use it for your robot, discuss your ideas with us so that we can provide the necessary guidance.

Good Luck! ☺

APPENDIX A

```
#include "Arduino.h"

void ledToggle()
{
  PORTB ^= 0b00100000;
}

void setup() {
  PRR &= 0b11111110;
  ADCSRA = 0b10000111;
  ADMUX = 0b01000000;
  DDRB |= 0b00100000;

}

void loop() {

    unsigned adcvalue, loval, hival;

    ADCSRA |= 0b01000000;

    while(ADCSRA & 0b01000000);

    ADCSRA |= 0b00010000;
    loval = ADCL;
    hival = ADCH;
    adcvalue = hival * 256 + loval;

    ledToggle();
    _delay_loop_2(adcvalue);

}
```

APPENDIX B

```
#include "Arduino.h"
#include <avr/interrupt.h>

unsigned int adcvalue;

void ledOn()
{
  PORTB |= 0b00100000;
}

void ledOff()
{
  PORTB &= 0b11011111;
}

ISR(ADC_vect)
{
      // Enter your solution here.
}

void setup() {

  PRR &= 0b11111110;
  ADCSRA = 0b10001111;
  ADMUX = 0b01000000;
  DDRB |= 0b00100000;

  sei();
  ADCSRA |= 0b01000000;

}

void loop() {

    ledOn();
    delay(adcvalue);
    ledOff();
    delay(adcvalue);

}
```

APPENDIX C

```
#include "Arduino.h"
#include <avr/interrupt.h>

unsigned int adcvalue;
unsigned int remapped_adc;

void InitPWM()
{
  TCNT0 = 0;
  OCR0A = 0;
  TCCR0A = 0b00000001;
  TIMSK0 |= 0b10;
}

void startPWM()
{
  TCCR0B = 0b00000100;
}

ISR(TIMER0_COMPA_vect)
{
      //  Insert your solution here
}

ISR(ADC_vect)
{
      //  Insert your solution here
}

void setup() {
  // put your setup code here, to run once:
  PRR &= 0b11111110;
  ADCSRA = 0b10001111;
  ADMUX = 0b01000000;
  DDRB |= 0b00100000;

  InitPWM();
  startPWM();
  sei();
  ADCSRA |= 0b01000000;

}

void loop() {}
```