Lab 5 Interfacing to Analog Inputs

– Analog Port C in ATmega328P

Objectives:

1. Understand the operation of the analog port C in ATmega328P.
2. Learn to program analog Port C in ATmega328P to obtain analog inputs.
3. Learn to interface with phototransistors and temperature sensors.

Lab Activities:

1. Interface with a temperature sensor

The LM35 series are precision integrated-circuit temperature sensors, with an output voltage linearly proportional to the Centigrade temperature with a 10 mV/ºC scale factor. LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature. The pin connection diagram is shown in Figure L5.1. Refer to the datasheet of LM35 for details.

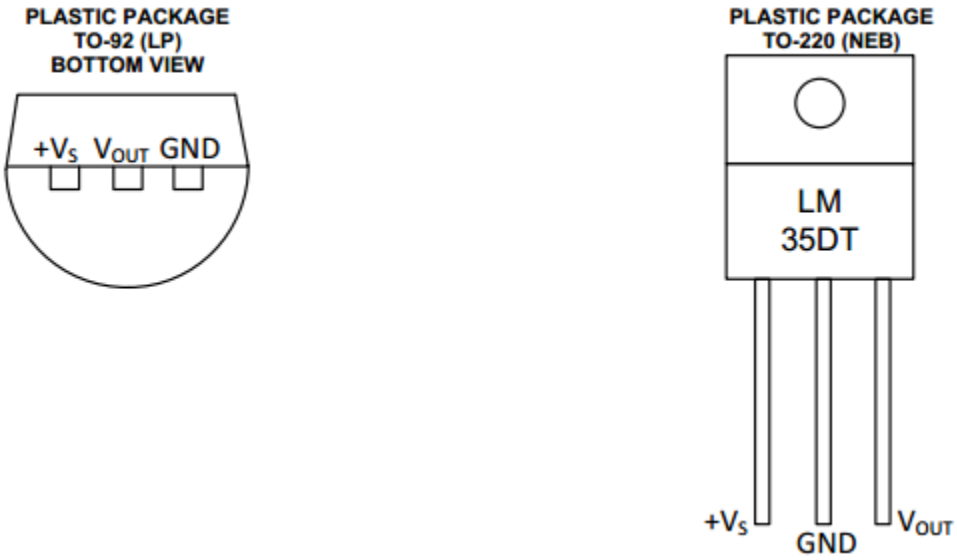

Figure L5.1 LM35 pin layout

In this part we will connect a linear temperature sensor LM35 to the analog port A0 and display the measured temperature in Serial Monitor. Connect LM35 to the Arduino board as shown in Figure L5.2. Run the program given in Example 1 as shown below.

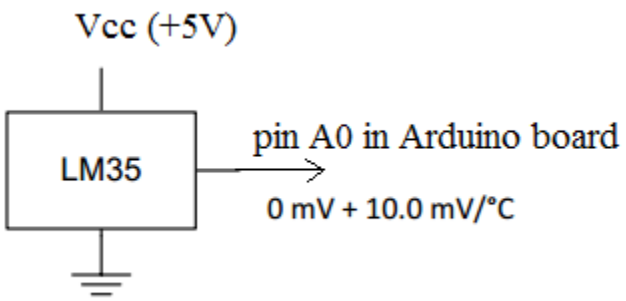

Figure L5.2 LM35 connection digram

```
/* Lab 5 ADC Example 1  Ver. 0
 * Display temperature in serial monitor
 * temperature sensor is connected to A0
 */
#include <AVR/io.h>
#define ADC_PreScaler_128  0x7

void setup()                                    // Built-in initialization block
{
  Serial.begin(9600);                           // Set data rate to 9600 bps

  /*REFS = 01 Internal 5V reference; ADLAR = 0  right adjusted;
  MUX3:0 = 0000   single-ended ADC0 */
  ADMUX = 0x40;
  /*ADCSRA: ADEN = 1 ADC enabled
            ADATE = 1 ADC Auto TRigger enable
            ADPS2:0 = 111  prescaler = 128  */
  ADCSRA |= ((1<<ADEN) | (1<<ADATE) | ADC_PreScaler_128);

  //ADCSRB: fADC auto Trigger Scoursce: free running mode
  ADCSRB = 0;
  //disable Digital input at ADC0 (i.e. A0)
  DIDR0 |= (1<<ADC0D);
}

void loop()                                     // Main loop auto-repeats
{
  char message[80];
```

```
  float volts;
  unsigned int adcread, temp;

  ADCSRA |= (1<<ADSC);                //start conversion
  while ((ADCSRA&(1<<ADIF)) == 0);    //wait till conversion is finished

  adcread = ADCL;   //read low byte first
  adcread = (((unsigned int) ADCH) << 8) + adcread;  //read high byte
  volts = (float(adcread))*5.0/1024.0;  //Vref = 5V
  temp = int (volts*1000);  //temperature in unit of 10 Celsius degree for
better accuracy in display

   sprintf(message, "\n A0 = %d, i.e. Temperature = %d.%d Celus degree",
adcread,  temp/10, temp%10);
  Serial.write(message);                  // Display voltage at A3
  delay(1000);                              // Delay for 1 second

}
```

After you run the program, you should be able to observe the results in the Serial Monitoring window.

Lab requirement 1:

Assume the room temperature range is between 15- 40 ºC, the analog output voltage from LM35 is between 0.15- 0.4 V. Therefore we can obtain the results in better accuracy if we choose 1.1 internal voltage as the reference voltage. Modify the program so you use internal 1.1V as the reference voltage and observe the difference. Show the demo to the instructor. Attach your code for Lab report. Please determine the temperature resolution (minimum temperature change that you can observe) for Vref = 5V and Vref = 1.1V, respectively.

2. Obtain relative light intensity level using a phototransistor

See the introduction in Lecture 7 for phototransistors. Connect the phototransistor as shown in Figure L5.3. The phototransistor in the Robotics Shield Kit is most sensitive to 850 nm wavelengths, which is in the infrared range. Many different light sources emit considerable amounts of it, including halogen and incandescent lamps and especially the sun. This phototransistor also responds to visible light but it's less sensitive. Run the program example 2. The analog voltage input in pin A3 is displayed the Serial monitor. Since the phototransistor lets more current pass when more light strike on its base, or less current pass with less light, the voltage at A3 increases as the light level increases. As smaller resistor will

make it more sensitive the light changes. You may need to use a larger resistor (such as 10 kΩ) for low ambient light condition.

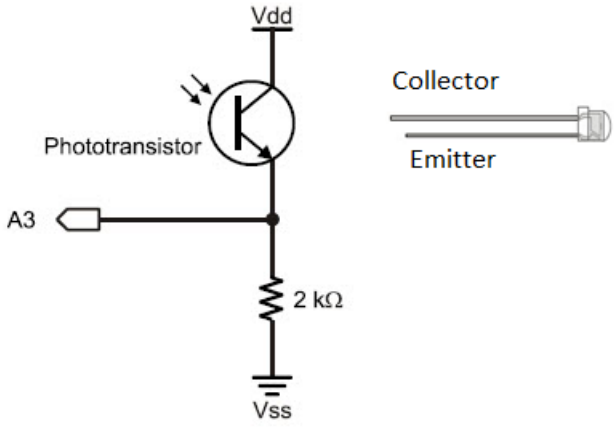

Figure L5.3 Phototransistor connection diagram

```
#include "avr/io.h"
/* Lab 5 Example 2
 * Robotics with the BOE Shield - Phototransistor
 * Display voltage of phototransistor circuit output connected to A3
 * in the serial monitor.
 */
#include <AVR/io.h>
#define ADC_PreScaler_128  0x7

void setup()                                  // Built-in initialization block
{
  Serial.begin(9600);                         // Set data rate to 9600 bps

  /*REFS = 01  Internal 5V reference; ADLAR = 0  right adjusted;
    MUX3:0 = 0011   single-ended ADC3 (pin A3 in Ariduino board) */
  ADMUX = 0x40 | 0x03;
  /*ADCSRA: ADEN = 1 ADC enabled
            ADATE = 1 ADC Auto TRigger enable
            ADPS2:0 = 111  prescaler = 128  */
  ADCSRA |= ((1<<ADEN) | (1<<ADATE) | ADC_PreScaler_128);

  //ADCSRB: fADC auto Trigger Scoursce: free running mode
  ADCSRB = 0;
  //disable Digital input at ADC3
  DIDR0 |= (1<<ADC3D);
```

```
}

void loop()                                   // Main loop auto-repeats
{
  char message[80];
  float volts;
  unsigned int adcread;

  ADCSRA |= (1<<ADSC);                  //start conversion
  while ((ADCSRA&(1<<ADIF)) == 0);     //wait till conversion is finished

  adcread = ADCL;                       //read low byte first
  adcread = (((unsigned int) ADCH) << 8) + adcread;  //read high byte
  volts= (float(adcread))*5/1024.0;     //convert to voltage, Vref = 5V
  sprintf(message, "\n A3= %d, i.e. analog input voltage = %d.%.3d V ",
adcread, int(volts), int(volts*1000)%1000); //in one line %.3d: display 3
digits with leading 0
  Serial.write(message);
  delay(1000);                           // update every 1 second
}
```

Lab Requirement 2:

Observe the analog voltage displayed in the serial monitor. Change the light level to observe the analog voltage changes.

3. Using an RC circuit to interfacing a phototransistor

In order to work with a wide range of light intensity and still sensitive to low light level, a RC circuit can be used to measure the discharging time, as shown in Figure L5.4 and run program example 3. The discharge time in microsecond is displayed in the serial monitor.

In the program, pin A3 is set as an output and output HIGH. The capacitor will be charged to 5V. Then we make pin A3 as an input, disconnect its pull-up resister and disable its digital input. The voltage is discharged through the phototransistor. If the light level is high, the phototransistor is able to discharge quickly, which leads to a small RC discharge time; while a larger discharge time when the light level is low.

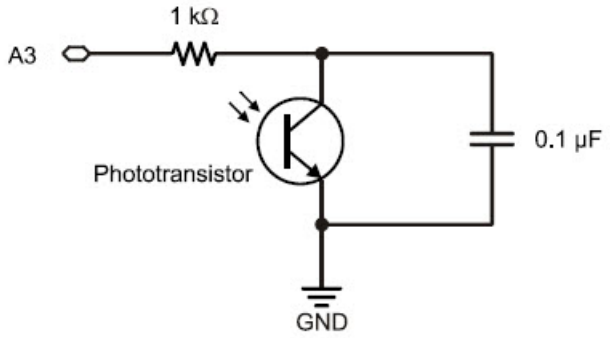

Figure L5.4 RC circuit using a phototransistor

```
/* Lab 5 Example 3
 * Robotics with the BOE Shield - Phototransistor connection using RC circuit
 * Display the discharge time in microseconds
 * in the serial monitor.
 */
#include <AVR/io.h>
#define ADC_PreScaler_128  0x7
#define ADC3 3
#define ADC4 4
unsigned int voltsLevel(char ADCpin);

void setup()                                  // Built-in initialization block
{
  Serial.begin(9600);                         // Set data rate to 9600 bps

  /*REFS = 01  Internal 5V reference; ADLAR = 0  right adjusted;
    MUX3:0 = 0011   single-ended ADC3 (pin A3 in Ariduino board) */
  ADMUX = 0x40 | 0x03;
  /*ADCSRA: ADEN = 1 ADC enabled
            ADATE = 1 ADC Auto TRigger enable
            ADPS2:0 = 111  prescaler = 128  */
  ADCSRA |= ((1<<ADEN) | (1<<ADATE) | ADC_PreScaler_128);

  //ADCSRB: fADC auto Trigger Scoursce: free running mode
  ADCSRB = 0;
  //disable Digital input at ADC3
  //DIDR0 |= (1<<ADC3D);
}
```

```
void loop()                                    // Main loop auto-repeats
{
  char message[80];
  float volts;
  unsigned int adcread[100];

  DDRC |= 0x08;   //set pin A3 (portC.3) as an output
  PORTC |= 0x08;  //output HIGH at pin A3
  delay(1);       //hold 5V for 1 ms to allow charging the capacitor
  DDRC &= ~0x08;  //set pin A3 as an input
  PORTC &= ~0x08;  //disconnect the pullup-resistor to allow discharge

  unsigned long time = micros();
  while (voltsLevel(ADC3)>10);  //wait till the read from ADC3 is less than 10
  time = micros() - time;
  sprintf(message, "\n rctime = %lu", time);
  Serial.write(message);
}

unsigned int voltsLevel(char ADCpin)
{
  unsigned int adcread;

  //disable Digital input at ADCpin
  DIDR0 |= (1<<ADCpin);

  ADCSRA |= (1<<ADSC);                 //start conversion
  while ((ADCSRA&(1<<ADIF)) == 0);     //wait till conversion is finished

  adcread = ADCL;                      //read low byte first
  adcread = (((int) ADCH) << 8) + adcread;  //read high byte
  return adcread;
}
```

Lab Requirement 3:

Change the light level to observe the RC time changes. In the above program, we can obtain different RC time by adjusting the level of voltage that we would like to discharged to, as shown in Figure L5.5.

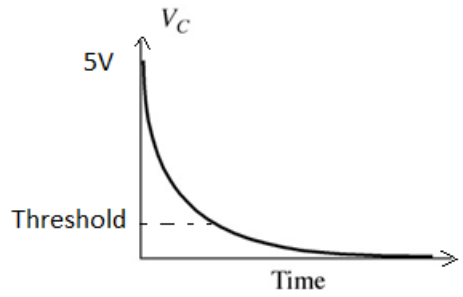

Figure L5.5 RC time

It can be achieved by adjusting the threshold level in the following program (highlighted in the example 3):

```
while (voltsLevel(ADC3)>10);
```

Modify the program and display the returned RC time for each threshold from 2 to 1022 in step of 2. Show the demo to the instructor. Choose a threshold value that you will use for the rest of lab.

Lab Requirement 4:

We will use two same phototransistor RC circuits for both side of robot and guide its movement according to the light level. Construct the circuit and adjust the phototransistors to point upward at a 45° from the breadboard, and outward about 90° apart, as shown in Figure L5.6.

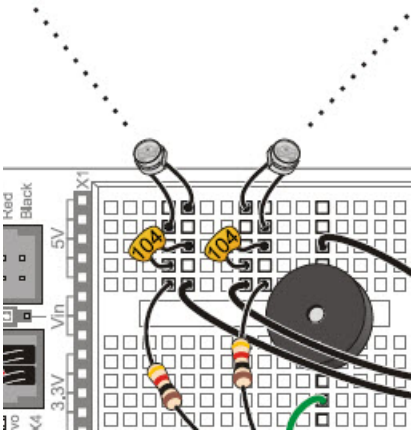

Figure L5.6 Phototransistor eye

Prepare a program so that both RC time from left and right side, named as **tLeft** and **tRight,** are displayed in Serial Monitor. You can construct a function to return the RC time to make your program more readable. The function prototype is given in the following:

unsigned long rcTime(char ADCpin, int threshold);

where the function will return the RC time in microsecond for given threshold and from the phototransistor connected to the analog pin ADCpin.

We may obtain the normalized light level difference from left and right side using the following code:

```
char ndShade;                        // Zeroed Normalized differential shade
ndShade =  (tRight)*100 / (tLeft + tRight) - 50;
```

We can see that the value of ndShade is between -50 to 50. if ndShade = 0, it means tRight is about the same as tLeft, which means both side has the same light level. If ndShade > 0, we have tRight > tLeft, it means left side is brighter than right side, i.e the shade is over right side phototransistor; otherwise ndShade < 0 , the shadow is over the left side.

Based on the value in ndShade, we may navigate the robot accordingly. When the shade is over left side, slow down the right wheel and the robot will turn to right side and move away from the shade. When the shade is over right side, slow down the left wheel and the robot will turn to the left side and move away from the shade. You may need to use the functions in Lab 2 & 3.

Complete a program so that your robot may navigate towards to the bright light. Use a flowchart to help you to design the sequential logic. After complete the program, show the demo the instructor. Please include the fully commented program in your report.

Report requirements:

Please follow the general requirements for Lab reports. The part in Lab requirement 4 needs to include both the flowchart and a complete circuit connection diagram. Please fully comment your programs.