Sean Webster

Microprocessors II

Lab 1 - Sensor Design and Analog Digital Conversion

Group 8 – Sean Webster, Eric Craaybeek, Munib Elkhatib

Handed in 10/3/2016 - Due 10/3/2016

**Personal contributions to the lab assignment:**

* Co-worked on getting threshold value working
* Provided some boilerplate code to work off of
* Created schematic
* Co-designed circuit

**Purpose:**

The purpose of this lab was to learn how to design sensors with embedded microcontrollers, understand the operation of ADC, and understand the design of sensor circuitry.

**Introduction:**

Sensors are the way to create physical input into microcontrollers from outside sources. Designing sensors and their associated circuity, and obtaining the data output by them are important tasks in designing an embedded system.

This lab was intended to design a light intensity sensor controlled by a PIC microcontroller. The parts included a PIC16F18857, a photoresistor, an LED, and some resistors. A breadboard was used to create the circuit. The ADC module in the PIC microcontroller was used to convert an analog signal to corresponding digital values.

The PIC based light sensor device:

1. Converted light intensity to a variable voltage.
2. Obtained an analog input signal via an analog pin of the PIC.
3. Used the internal ADC of PIC to convert analog signals to digital values.
4. Turned on an LED if the sensor is encapsulated in a dark box or put in a dark environment.
5. Turned off the LED if the sensor was placed in a well-lit environment.

**Materials, Devices, and Instruments:**

* PIC16F18857 Microcontroller
* Photoresistor
* LED
* 10k ohm and 220 ohm resistors
* PIC Kit 3
* Breadboard + jumpers

**Schematic:**

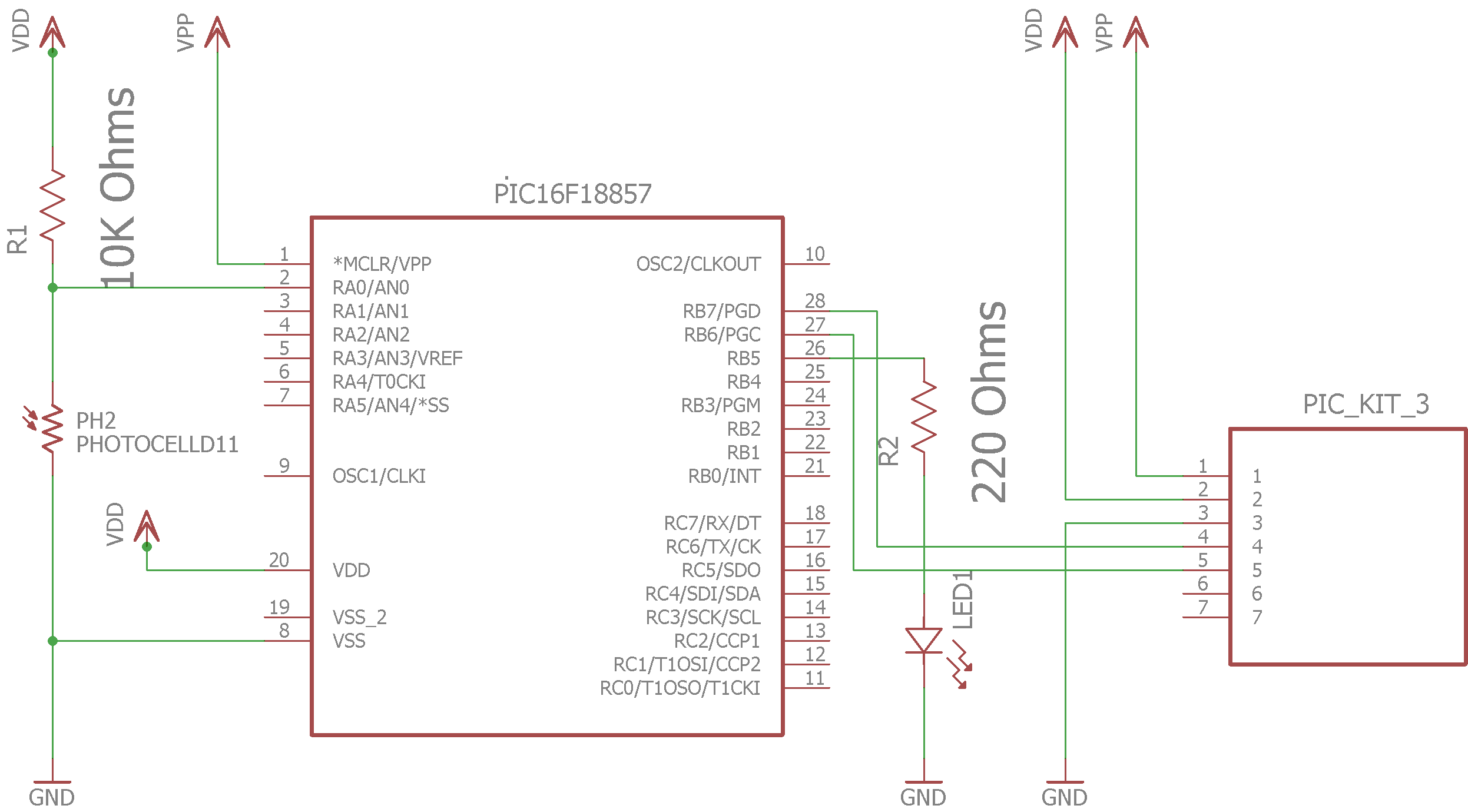


Figure 1 – Schematic of whole lab.

**Lab Methods and Procedure:**

C:\Users\Sean\Downloads\Micro2Lab1.png

Figure 2 – program flowchart.

1. Set up main program
   1. Define running frequency
   2. Set port B to outputs and bit 5 to digital
   3. Set port A to inputs and bit 1 to analog
   4. Initialize working variable
2. Main program loop
   1. Wait for ADC to grab value
   2. Shift low and high registers of ADC value into variable
   3. Conditional statements based on threshold number to turn led on or off
   4. Repeat loop

The most important part of this program was the ADC and DAC. This was done by first setting up the ports for each, and then using a delay to wait for ADGO to grab a value from the right ports. Those values were then checked, and then a value was output to the output ports.

**Troubleshooting:**

* Problem: Light wasn’t turning on.
  + Solution: The ADCON0 bit 7 needed to be turned on to turn the whole adc on. This was not included in any sample code.
* Problem: PIC Kit 3 not providing power.
  + Solution: IDE has a setting that has to be enabled for PIC Kit to provide power.
* Problem: Threshold value makes no difference.
  + Solution: Value is default left justified. By making it right justified, it is easier to manipulate.

**Results:**

* Moving hand over photoresistor caused light to turn on.
* Removing hand caused light to shut off.

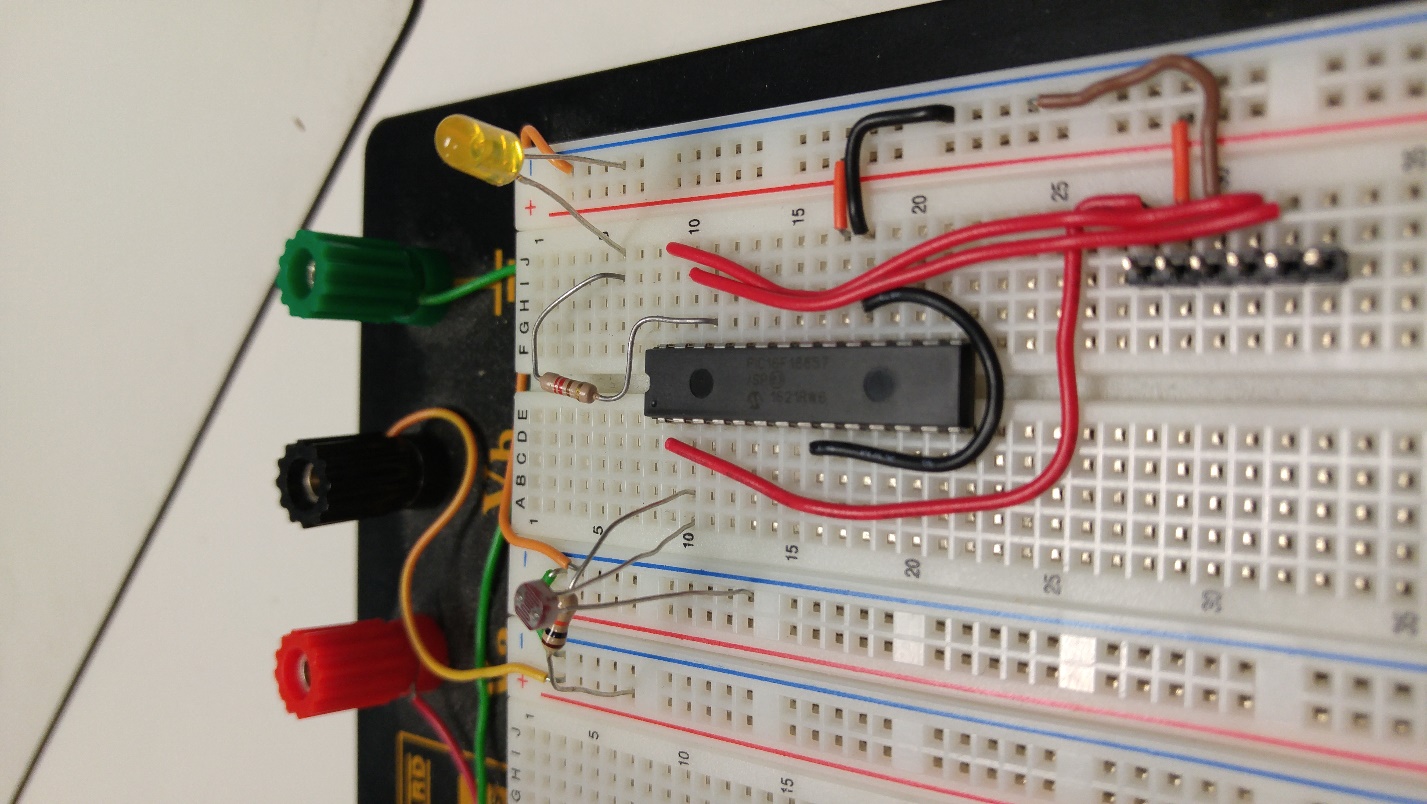
****

Figure 3 – Picture of circuit with light off

**Contributions to this lab:**

* Eric Craaybeek
  + Co-created circuit
  + Co-coded program
  + Co-Worked on ADC
  + Reviewed PIC manual
* Munib Elkhatib
  + Co-worked on ADC
  + Co-coded program
  + Provided boiler plate code
  + Reviewed PIC manual
* Sean Webster
  + Co-created circuit
  + Co-coded program
  + Reviewed PIC manual
  + Provide boiler plate code
  + Created schematic

**Appendix:**

/\*

\* File: main.c

\* Author: Eric Craaybeeek \*

\* Created on September 21, 2016, 4:40 PM

\*/

//#define \_XTAL\_FREQ 500000

#include <xc.h>

#include <htc.h>

void main(void) {

//set port B to outputs and bit 5 to digital

TRISB = 0x00;

ANSELBbits.ANSB5 = 0;

//set port A to inputs and bit 1 to analog

TRISA = 0xFF;

ANSELAbits.ANSA0 = 1;

//set up ADC

ADCON1 = 0b11110000;

ADCON0 = 0b10010001;

ADPCH = 0b00000000;

ADREF = 0b00000000;

//initialize working variable A

int A = 0;

while(1)

{

//wait until ADC is finished

while (ADCON0bits.ADGO)

{

\_\_delay\_ms(1);

}

//put 10 digit output into A

A = (ADRESH << 8);

A = A + ADRESL;

//conditional to turn on LED based on A

if (A > 0b111)

{

LATBbits.LATB5 = 0;

}

else

{

LATBbits.LATB5 = 1;

}

//restart ADC

ADCON0bits.ADGO = 1;

}

return;

}