Application Note: Arduino Microcontroller

By **Baban Malhi**

Since my major is in Computer Engineering and Electrical Engineering and also has embedded system job position, I decided to work on the Arduino Microcontroller.
I also supported team Oreo with other task such as PCB design and modular but I decided to make lab report on Arduino since other member of the team will cover PCB design.

Introduction: Arduino is one of the most popular physical computing platforms available today. It's an amazing tool for both experienced and beginner electronics students and electronic experts. Since it is very easy to use, a lot of schools are asking students to use it, which is why it is getting very popular. Its hardware and software, both combines and make a well supportive and solidly designed electronic platform. The best thing about the Arduino is, it is an open source which means the schematics, hardware design files, and source code are all freely available for viewing and modification to the world. Which means, the license to Arduino is free, anyone can download the schematics and modify it to make microcontroller according to how they want it to work for them. So everyone is free to make changes on the hardware design and produce their own version. The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller, simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. Reference design and Schematics of the microcontroller can be found at the end of this report in Appendix at the last page of this report. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX).

Input and Output: This is the very important part of the lab because I had problem figuring out which ports used for what so I will explain it for you guys, Each of the 14 digital pins on the Arduino Uno can be used as an input or output. Arduino has its own system call function such as pinMode(), digitalRead() and digitalWrite().They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip. Please see **appendix C** for the pin diagram for ATmega328).

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite()

function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provides 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analogReference().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields, which block the one on the board.

USB over current Protection: The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

How Arduino was used in RF Project: In out labs, we started with building a simple function generator that can output triangle, square, sawtooth, and sinusoidal waveforms. We used direct digital synthesis circuits, DDS circuits. A DDS circuit takes advantage of the ever-increasing speed of digital circuits to implement fast waveform generators. This is where Arduino UNO comes handy because we used Arduino UMO 8-bit microcontroller with an ATmega328P chip to emulate a DDS circuit. We used to ATmega328P to program, which we used later on our PCB for the final product.

Do you have everything you need?

- You need the Arduino UNO itself.
- A USB cable to connect the Arduino with the computer to program, it is standard A-B cable, any length is OK

Lets get started:

- Take the Arduino out of its protective bag. (if there is anything missing, make sure to contact your professor or the retail store where you got the hardware from) Make sure it looks like the picture below.
- To power it on, plug one end of USB cable into the Arduino and the other end into your computer. You computer will power the Arduino at this point since you are not using the external power (Make sure Arduino is connected straight to computer, some keyboard also have USB port- do

not connect to those ports because sometimes those ports will not allow you to program the Arduino which is why I recommend you to connect straight to your computer for saving time)

- If you connect the Arduino right, you should see a green light on the right side of Arduino and also an orange light. (if no light blinks, Make sure your connections are tight and also your computer is on and please redo all the steps you done so far otherwise your hardware is not working properly and that case get help from your professor or from the retail store you got the hardware from)

Lets make your Arduino do something: We will start with "Hello World". In this part of tutorial, you will learn the basics of Arduino, using Arduino software and uploading sketch to the Arduino board.

- Go to Arduino software download page and download the most updated version of the software.
- Extract the package onto the desktop and run it to install the software onto your computer
- Double click the Arduino software icon to open up the workspace, you can find it in "start-> Programs-> Arduino.exe"
- Configure the chip you are using, in our case it should be ATMEGA328P but to be sure, you can check it by reading the text on the chip it should say ATMEGA plus some number.
- One you configure the chip, go to your Arduino application and click on the "tools -> microcontroller (MCU) -> atmega328p", if you using different chip, you can select that in this class but make sure you select the right chip. (your preferences will be saved, you will not have to do this step again unless you decide to use different chip
- Select the port: to do that, in Arduino software select "tools -> Serial Port -> COMX" make sure you select COM it is arbitrary so you can go to serial ports in control panel and find what COM is attached to your Arduino and select that com.
- Open Blink Sketch: Sketches are little scripts that you can send to the Arduino to tell it how to act. Let's open up an Example Sketch. Go to the File menu -> Sketchbook -> Examples -> Digital -> Blink
    - The window should now show a bunch of text in the formerly empty white space and the tab **Blink** above it
- The first step to getting a **Sketch** ready for transfer over to the Arduino is to **Verify/Compile** it. It is basically checking for any errors, if not then it will build the program which means it is converting your code to the language, which Arduino understands, and if everything works find then it should give no error and after few seconds, you should see the message "Done compiling" No error means it is ready to upload to the Arduino board.

- Reset: you must press the rest button on the board to prepare Arduino for the new upload, this way your hardware is looking forward to flash the new build for the board. To do it, simply press the black button on the board.
- Upload: Lets do what we been waiting for, To upload, simply click on file and the “Upload to I/O Board and after few seconds, you should get new screen with a message “Done Uploading”.
    - If you get the following error message "**avrdude: stk500_getsync(): not in sync: resp=0x00**" that means that the Arduino is not responding
    - If you get the following error: **can't open device "COM10": The system cannot find the file specified.** It means that you have too many COM ports (maybe you've got 9 Arduinos?) You should make sure that the port is numbered as low as possible.

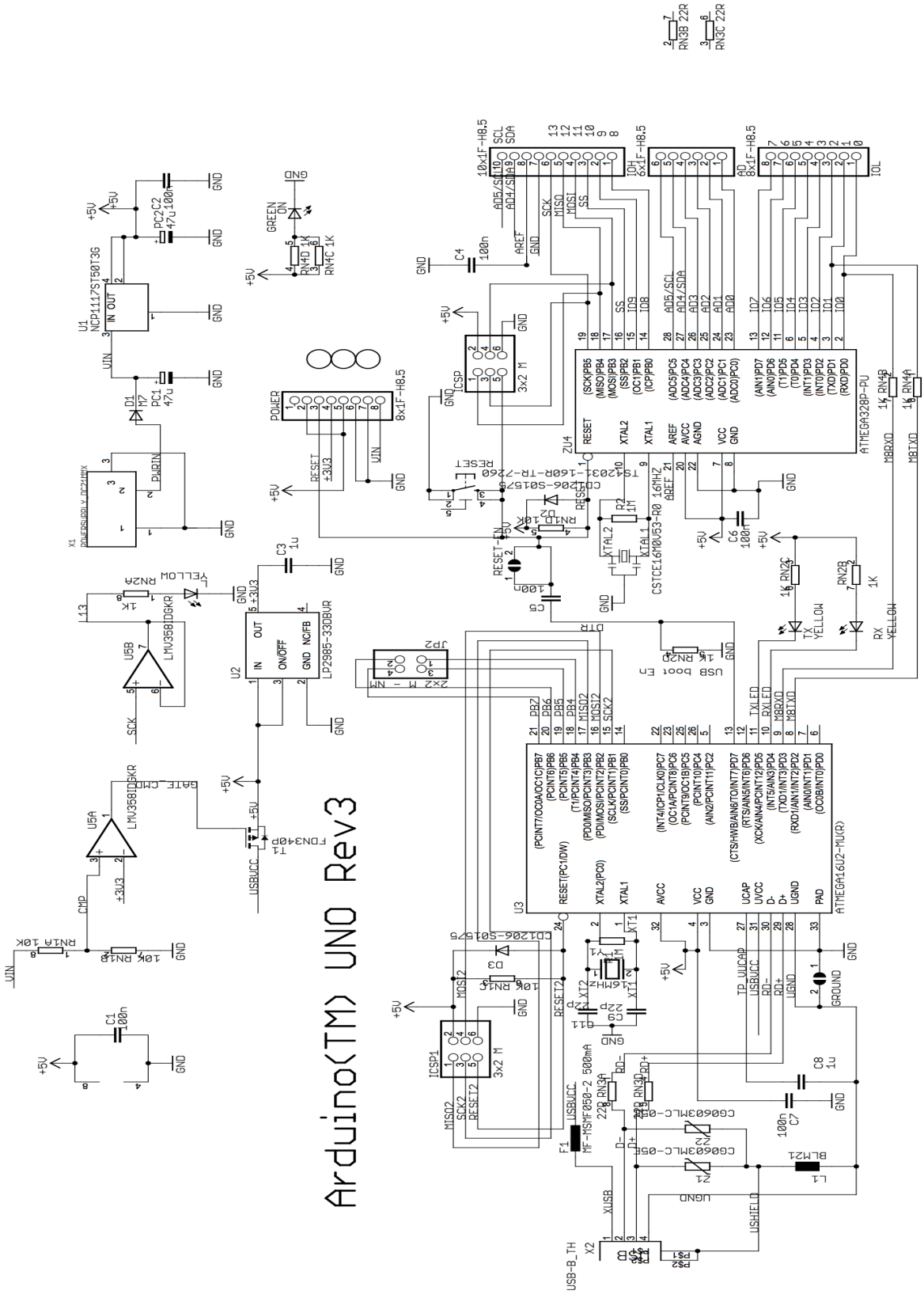
Notice: The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

APPENDIX:

Appendix A:
Tutorials on Arduino Coding
https://www.youtube.com/playlist?list=PLA567CE235D39FA84
http://arduino.cc/en/uploads/Main/arduino_Uno_Rev3-02-TH.zip

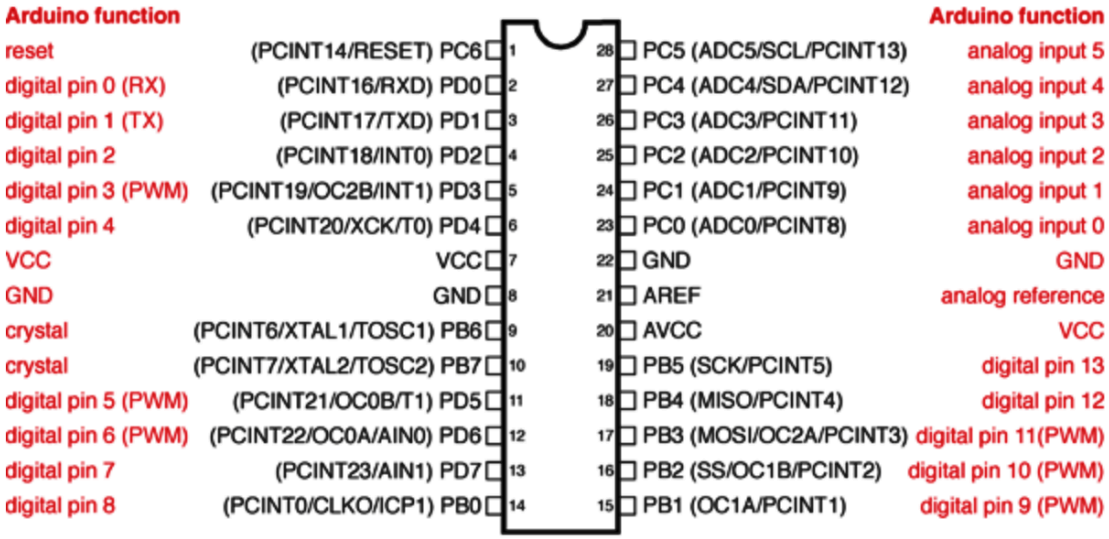
Appendix B:
Citation: http://www.arduino.cc
http://www.atmel.com
http://en.wikipedia.org/wiki/ATmega328
http://en.wikipedia.org/wiki/Arduino

Appendix C:
Schematic of the microcontroller:



Appendix D

Atmega168 Pin Mapping



Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.