MICHIGAN STATE UNIVERSITY COLLEGE OF ENGINEERING

Programming the Arduino Leonardo

Using USB Libraries

Tyler James Dennis

4/5/2013

Tyler Dennis

ECE 480 Team 5

The following application note gives the reader the basic knowledge required to begin programming an Arduino Leonardo using USB libraries. The document describes the steps to set up the Arduino Leonardo, describes the USB library functions, and walks the reader through an example program.

Table of Contents

Introduction to Arduino

An Arduino is a microcontroller. The Arduino line of microcontrollers was designed specifically to streamline the creation of electronic prototypes for hobbyists and small design projects. The Arduino is programmed using the Arduino programming language, which is based on the Wiring programming language. The Arduino software uses a development environment which is based on the existing IDE, Processing.

Arduino Leonardo

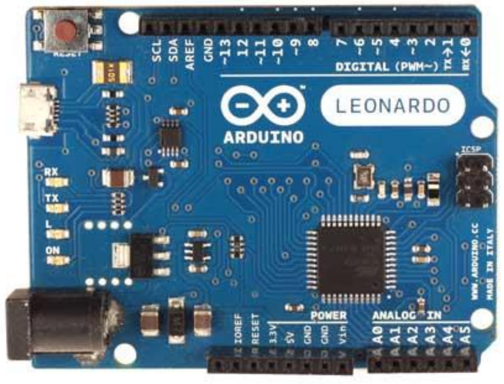


Figure 1. The Arduino Leonardo

Unlike the other Arduino boards, the Arduino Leonardo has a single microcontroller for the USB connection and the user created Arduino programs. This gives the user much more flexibility when it comes to a computer's recognition of the device. The Leonardo also offers more functionality as an input device than other Arduino models. This is due to the fact that it is recognized as an input device upon connection to the computer through a USB port. The Leonardo has 20 input/output pins, which give the user plenty of inputs and outputs for maximum functionality. The board also has a built in micro-USB port and power jack to support the use of an external power supply.

Getting Started with the Arduino Leonardo

In order to begin using the Leonardo, the user needs the following items:

1. Personal Computer
2. Micro-USB Cable

3. Arduino Leonardo

1) Download the Arduino IDE

The Arduino IDE is the environment in which the Arduino Leonardo can be programmed. This software is a necessary tool for the developer. The Arduino IDE can be downloaded from the Arduino website. A link has been provided below:

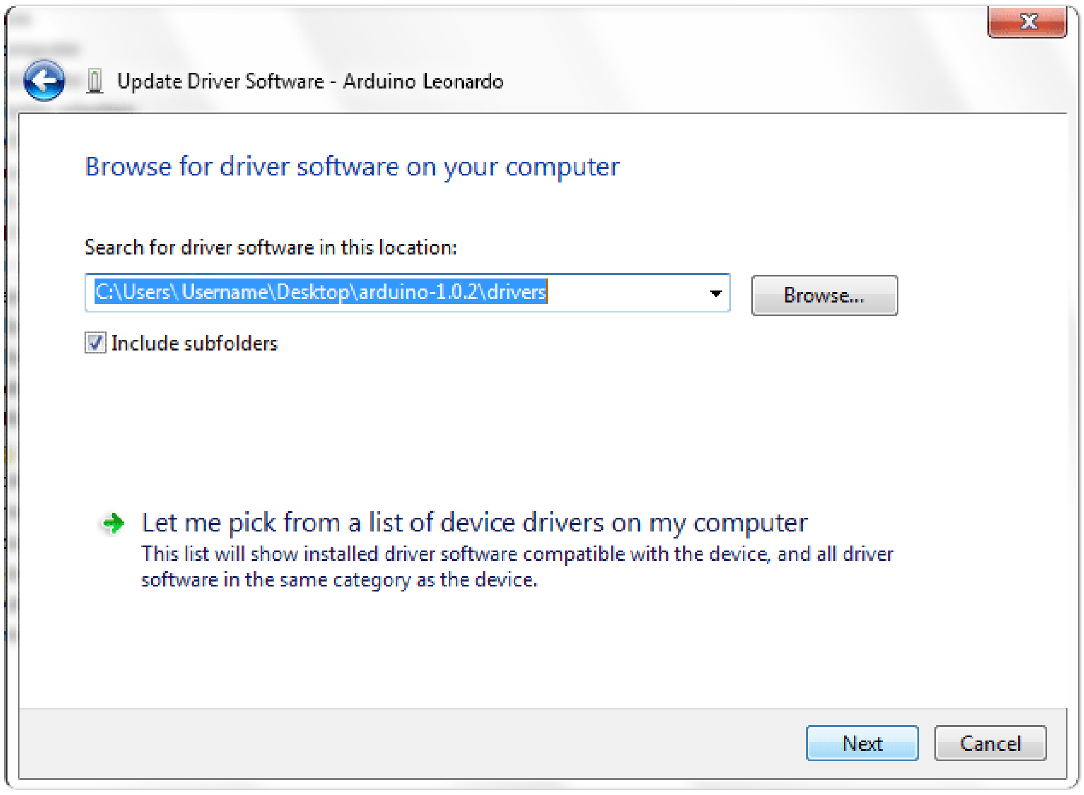
http://arduino.cc/en/Main/Software

2) Connect the Board to the Computer

The board needs to be connected to a computer using a micro-USB cable. Once connected, the Arduino Leonardo will begin receiving power through the USB connection. The green LED labeled "ON" will light up.

3) Download the Arduino Drivers

The latest Arduino drivers can be located and downloaded through the Windows device manager. Simply right click on the Arduino Leonardo device in the list, and select update driver. The user needs to ensure that the drivers are downloaded to the proper folder. Figure 2 shows the proper directory:



Update Driver Software - Arduino Leonardo

Browse for driver software on your computer

Search for driver software in this location:

C:\Users\ Username\Desktop\arduino-1.0.2\drivers

☑ Include subfolders

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Figure 2. Arduino Driver Installation Directory

4) Choose the Preferred Device

In order to include the proper libraries, the user needs to choose to use the Arduino Leonardo through the Arduino IDE. In order to do this the user needs perform the following within the IDE:

Select Tools -> Board -> Arduino Leonardo

Once steps 1 through 4 have been completed, the Arduino Leonardo can be programmed.

USB Library Functions

The Arduino Leonardo is one of the few Arduino boards that can be programmed using USB libraries. These libraries allow the user to easily emulate a keyboard or mouse using the Arduino Leonardo. The USB libraries are automatically included in the program when the user chooses to use the Arduino Leonardo through the Arduino IDE. Some of the basic keyboard and mouse emulation commands are explained below.

Mouse

Mouse.begin() and Mouse.end()

The begin() function initiates the mouse library. This allows the Leonardo to begin functioning as a mouse. Conversely, the end() function stops the emulation.

Mouse.click()

The click() function emulates a mouse click. The function defaults to a left mouse button click. However, the click() function can accept three different parameters:

- MOUSE_LEFT – Clicks the left mouse button
- MOUSE_RIGHT – Clicks the right mouse button
- MOUSE_MIDDLE – Clicks the middle mouse button

Mouse.move()

The move() function allows the Leonardo to control the location of the mouse cursor on the screen. The move() function requires three integer arguments:

- xVal – Defines the amount the cursor moves in the x-axis
- yVal – Defines the amount the cursor moves in the y-axis
- wheel – Defines the amount the scroll wheel scrolls

Mouse.press() and Mouse.release()

The press() function operates similarly to the click() function. However, the press() function holds the chosen button until it is released manually with the release() function. The press() and release() buttons both default to the left mouse button, but can accept the same arguments as the click() function.

Mouse.isPressed()

The isPressed() function returns a Boolean (true or false) if a given button is pressed. This function also defaults to the left mouse button, and can accept the same arguments as the press(), release(), and click() functions.

Keyboard

Keyboard.begin() and Keyboard.end()

The begin() and end() functions of the keyboard library operate in the same fashion as their counterparts in the mouse library.

Keyboard.press() and Keyboard.release()

The press() and release() functions also operate similarly to their corresponding functions in the mouse library. However, the user can specify which key is pressed or released by passing a character argument.

Keyboard.print()

The print() function is used to send keystrokes to the computer. As parameters, the function accepts characters. Characters can be passed either one at a time or as strings.

Keyboard.println()

The println() function operates similarly to the print function. However, the println() function always performs a carriage return and starts a new line at the end of the printed characters.

Keyboard.releaseAll()

The releaseAll() function operates similiarly to the release() function, but it releases all pressed keys at once.

Keyboard.write()

The write() function allows the user to send any keystroke to the computer, including non-character keystrokes. Examples of non-character keystrokes include the backspace, control, alt, and function keys. The write() function can accept integer (ASCII), character, hex, and binary arguments.

Example Code

This section will showcase and explain (using comments) an example program that features the use of the keyboard and mouse libraries.

```
// Initializes the analog pins on the Arduino and assigns each to a corresponding keyboard key
const int upButton = 2;
const int downButton = 3;
const int leftButton = 4;
const int rightButton = 5;
const int mouseButton = 6;

void setup()
 {

        //Designates the analog pins to be inputs
         pinMode(upButton, INPUT);
         pinMode(downButton, INPUT);
         pinMode(leftButton, INPUT);
         pinMode(rightButton, INPUT);
         pinMode(mouseButton, INPUT);

         Serial.begin(9600);
        // initializes the mouse and keyboard libraries
        Mouse.begin();
        Keyboard.begin();
}

void loop()
 {
 // Instructs the Arduino to use the serial port for mouse control
 if (Serial.available() > 0)
 {
         char inChar = Serial.read();
          switch (inChar)
```

```
            {
           //The following cases move the mouse cursor based on the data stored in the inChar variable
            case 'u':
           Mouse.move(0, -40);
           break;
           case 'd':
            Mouse.move(0, 40);
            break;
           case 'l':
            Mouse.move(-40, 0);
            break;
            case 'r':
           Mouse.move(40, 0);
            break;
           case 'm':
           Mouse.click(MOUSE_LEFT);
           break;
           }
 }

 // Keystrokes are emulated when the chosen analog input pin reads high
 if (digitalRead(upButton) == HIGH)
           {
           Keyboard.write('u');
           }
 if (digitalRead(downButton) == HIGH)
           {
           Keyboard.write('d');
           }
 if (digitalRead(leftButton) == HIGH)
            {
           Keyboard.write('l');
           }
 if (digitalRead(rightButton) == HIGH)
            {
           Keyboard.write('r');
           }
 if (digitalRead(mouseButton) == HIGH)
            {
           Keyboard.write('m');
           }
}
```

In summary, this program simulates a keypress and moves the mouse cursor simultaneously when a non-zero voltage is detected at any of the declared analog input pins. If momentary switches are wired to control the voltage of these pins, then the switches can trigger the keyboard and mouse functions. The practical applications of this program are virtually nonexistent, but it shows how both of the libraries work.

Summary

The Arduino Leonardo offers many tools that streamline the emulation of input devices. The keyboard and mouse libraries and built in micro-USB port are at the core of this convenience. There are also user-created libraries that can be downloaded to enhance the performance of the Leonardo even further. The simplicity and power of the Leonardo allow for great opportunities in the creation of input devices.

References

- Arduino Main Page
    - http://www.arduino.cc/
- Arduino Leonardo USB Libraries
    - http://arduino.cc/en/Reference/MouseKeyboard
- Arduino Software Download
    - http://arduino.cc/en/main/software