Using an Arduino as a Slave Board

Arduino serial interfacing and basic function library

Table of Contents

By Mike Molianri

Home automation, Robots, Vending machines. What do they have in common?

The answer is computers controlling hardware. Making lights go on and off. Controlling motors, relays and other electronic components.
Liberty Basic for windows makes programming simple but lacks the ability to directly control or interface with hardware (except the parallel port).

To make hardware interface coding simpler a set of functions is required to facilitate the communication with external hardware. Because the arduino is the most widely used hobby micro controller in the world and has so much support it is the obvious choice to make a set of simple functions for LB users to take advantage of this great platform for there projects.

If you are a complete novice and have no Arduino experience you might check out this beginners article.
http://lbpe.wikispaces.com/Fun+with+the+Arduino

The things covered in this article are limited to simple port I/O (Input and output).

- Test if pin is high or low
  - Get input from buttons and other sensors connected to the arduino
- Set a pin high or low
  - Turn things on and off like lights, buzzers, motors, ect.
- PWM (Pulse width modulation) output
  - Dim lights, Control servo motors, interface with variable speed motor controllers
- PWM (Pulse width modulation) input
  - Get input from things like sliders and position information

Getting Started with the functions

Code Example Circuit

-Initialization code

This code sets up the arrays that will be used to manage pin settings and values.

Should be placed at start of program.

```
dim ad.pinstatus(10,100)dim ad.pinS
etting$(10,100)for x = 0 to 10    f
or y = 0 to 100       ad.pinSetting
$(x,y) = "Un-Set"  next ynext x
```

-Setting a pin Status

To set a pin high or low you can call the following

function.
The com port must be specified as a number (To address com3 you would use 3)
The pin must be a number of a pin present on the arduino.
The value to send must be a 1 or 0 (1 = high, 0 = low)
The return value will be a 1 if successful and 0 if not.

```
return.value = AD.Set(Com.port, Pin
, Val.To.Send)
```

-Pulling the input status of a pin

This function will retrieve a 1 or 0 depending on what the status of a sensor connected to the arduino is.
The com port must be specified as a number (To address com3 you would use 3)
The pin must be a number of a pin present on the arduino.
Will return a 1 or a 0 depending on if the input pin is pulled high or low.

```
return.value = AD.Get(Com.port.Sele
ction, Pin.Selection)
```

-Setting a PWM output

PWM Will allow you to control a servo, dim a light, ect.
The com port must be specified as a number (To address com3 you would use 3)
The pin must be a number of a pin present on the arduino.
The value must be a number between 0 and 254.

```
return.value = AD.PWM.Out(Com.port.
Selection, Pin.Selection, Val.To.Se
nd)
```

-Pulling the PWM input status of a pin

The com port must be specified as a number (To address com3 you would use 3)
The pin must be a number of a pin present on the

arduino.
The return value will be number between 0 and 254.

```
return.value = AD.PWM.In(Com.port.S
election, Pin.Selection)
```

-Setting a servo output

Will allow you to control a servo.
The com port must be specified as a number (To address com3 you would use 3)
The pin must be a number of a pin present on the arduino.
The value must be a number between 0 and 180, the number of degrees the servo will rotate to.

```
return.value = AD.Servo(Com.port.Se
lection, Pin.Selection, Val.To.Send
)
```

-Managing Pin Settings

There are 2 arrays that manage what a pin is set to (set, get, pwm.in, pwm.out, servo) and what the last value assigned to that pin was or retrieved from the device as an input pin.

With these arrays to can both retrieve the current pin value and setting. You can also set the value and pin setting. In this way you can update the values and settings for a group of pins and apply those settings all at once with the

```
value.pin.was.set.to = ad.pinstatus
(Com.port, Pin)
```

-Set and retrieve all pin settings and values at once

The AD.Update.All() function will use the 2 arrays that are defined at the start of the program and update the values of each pin based on the inputs from the arduino and also set the output values on the arduino device.

This allows for all the pins of all the connected arduino devices to be updated to the values from the ad.pinstatus() array and the pin type setting from the ad.pinSetting$() array.

```
bla = AD.Update.All()
```

-Closing your program gracefully closing all the com ports

This function will close all the open com ports as a com port is kept open after any interaction with the arduino module.

```
AD.close.all.the.com.ports()
```

The Arduino Code

To make liberty basic talk with an arduino they have to speak the same language.
In this case a string is used and sent over to the device followed by a carriage return character CHR$(13)

The only thing you really have to understand about the arduino code is that it must be download to the arduino board using the arduino IDE.
This can be download from the following link.
http://arduino.cc/en/Main/Software

Note: This code is written in the Arduino's special form of c and is not basic code.

```
#include <Servo.h>

// Buffer to store incoming commands from serial port
String inData;
```

```
//declare each of the servos as a variable of type servo
     Servo LBServo1;
     Servo LBServo2;
     Servo LBServo3;
     Servo LBServo4;
     Servo LBServo5;
     Servo LBServo6;
     Servo LBServo7;
     Servo LBServo8;
     Servo LBServo9;
     Servo LBServo10;
     Servo LBServo11;
     Servo LBServo12;

void setup() {
  Serial.begin(9600);
  Serial.println("Serial conection started, waiting for instructions..
.");
}

void loop() {
  while (Serial.available() > 0)
  {
    char recieved = Serial.read();
    inData += recieved;

    // Process message when new line character is recieved
    if (recieved == '\n')
    {
      String Param0 = getValue(inData, ' ', 0);
      String Param1 = getValue(inData, ' ', 1);
      String Param2 = getValue(inData, ' ', 2);
      String Param3 = getValue(inData, ' ', 3);
      String Param4 = getValue(inData, ' ', 4);
      String Param5 = getValue(inData, ' ', 5);

      inData = "";

      int valParam0 = Param0.toInt();
      int valParam1 = Param1.toInt();
      int valParam2 = Param2.toInt();
      int valParam3 = Param3.toInt();
      int valParam4 = Param4.toInt();
      int valParam5 = Param5.toInt();
```

```
//Serial.print(valParam0);
//Serial.print(valParam1);
//Serial.print(valParam2);
//Serial.print(valParam3);
//Serial.print(valParam4);
//Serial.print(valParam5);



if ( Param0 == "get")
{
  pinMode(valParam1, INPUT);
  Serial.println(digitalRead(valParam1));


}

if ( Param0 == "set")
{
  pinMode(valParam1, OUTPUT);
  digitalWrite(valParam1, valParam2);
  Serial.println("1");


}


if ( Param0 == "pwm.out")
{
  analogWrite(valParam1, valParam2);
  Serial.println("1");

}



if ( Param0 == "pwm.in")
{
  Serial.println(analogRead(valParam1));
}

if ( Param0 == "servo")
{
    switch (valParam1) {
```

```
case 1:
        LBServo1.attach(1);
        LBServo1.write(valParam2);
        Serial.println("1");
        break;
case 2:
        LBServo2.attach(2);
        LBServo2.write(valParam2);
        Serial.println("1");
        break;
case 3:
        LBServo3.attach(3);
        LBServo3.write(valParam2);
        Serial.println("1");
        break;
case 4:
        LBServo4.attach(4);
        LBServo4.write(valParam2);
        Serial.println("1");
        break;
case 5:
        LBServo5.attach(5);
        LBServo5.write(valParam2);
        Serial.println("1");
        break;
case 6:
        LBServo6.attach(6);
        LBServo6.write(valParam2);
        Serial.println("1");
        break;
case 7:
        LBServo5.attach(7);
        LBServo7.write(valParam2);
        Serial.println("1");
        break;
case 8:
        LBServo8.attach(8);
        LBServo8.write(valParam2);
        Serial.println("1");
        break;
case 9:
        LBServo9.attach(9);
        LBServo9.write(valParam2);
        Serial.println("1");
        break;
case 10:
```

```
                    LBServo10.attach(10);
                    LBServo10.write(valParam2);
                    Serial.println("1");
                    break;
            case 11:
                    LBServo11.attach(11);
                    LBServo11.write(valParam2);
                    Serial.println("1");
                    break;
            case 12:
                    LBServo12.attach(12);
                    LBServo12.write(valParam2);
                    Serial.println("1");
                    break;
          }


      }

    }
  }
}




String getValue(String data, char separator, int index)
{
  int found = 0;
  int strIndex[] = {
    0, -1
  };
  int maxIndex = data.length() - 1;
  for (int i = 0; i <= maxIndex && found <= index; i++) {
    if (data.charAt(i) == separator || i == maxIndex) {
      found++;
      strIndex[0] = strIndex[1] + 1;
      strIndex[1] = (i == maxIndex) ? i + 1 : i;
    }
  }
  return found > index ? data.substring(strIndex[0], strIndex[1]) : ""
;
}
```

The Liberty Basic Demo

The demo program shows off the basic functionality of the library.

The Com port that the arduino is on must be placed in the com port number text box.

A pin must be selected and a value to send must be entered.

The 4 buttons on the right are used to execute the functions.

The return value is populated by the what is returned by the arduino.

```
    dim ad.pinstatus(10,100)
    dim ad.pinSetting$(10,100)

    for x = 0 to 10
        for y = 0 to 100
            ad.pinSetting$(x,y) = "Un-Set"
        next y
    next x
```

```
 'Populate pin list
dim  pins$(100)
 for x = 0 to 13
    pins$(x) = str$(x)
 next x
```

```
'Populate com port drop down
dim Comm.Ports$(10)
 for x = 0 to 10
    Comm.Ports$(x) = str$(x)
 next x

    'nomainwin
```

```
    WindowWidth = 450
    WindowHeight = 445
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)

    listbox #main.PinSelector, pins$(, [listbox1DoubleClick],   10,  8
2, 40, 320
    listbox #main.PinSetting, pin.Settings$(, [set.pin.settings],   50
,  82, 40+70, 320
    listbox #main.PinStatuses, pin.statuses$(, [set.pin.output],   90+
70,  82, 50, 320
    statictext #main.statictext2, "Pin",  10,  62, 40,  20
    statictext #main.statictext50, "Setting",  50,  62, 40,  20
    statictext #main.statictext55, "Value",  90+70,  62, 40,  20


    textbox #main.ValToSend, 165+70,  32, 170,  25
    statictext #main.statictext4, "Value To Send", 165+70,  12, 170,
20
    statictext #main.statictext5, "Return Value", 165+70,  62, 175,  2
0
    textbox #main.ReturnValue, 165+70,  82, 170,  25
    button #main.button7,"Set Pin High/Low Status",[Set.Pin], UL, 165+
70, 117, 170,  25
    button #main.button8,"Get Pin High/Low Status",[Get.Pin], UL, 165+
70, 152, 170,  25
    button #main.button9,"PWM Output",[PWM.Out], UL, 165+70, 187, 170,
  25
    button #main.button10,"PWM Input",[PWM.In], UL, 165+70, 222, 170,
 25
    button #main.button11,"Servo Out",[Servo.out], UL, 165+70, 222+35,
 170,  25

    button #main.button11,"Update All Pins",[Update.Pins], UL, 165+70,
 300, 170,  25
    statictext #main.statictext11, "Com Port Number",  10,  12, 145+70
,  20
    combobox #main.ComPortNo, Comm.Ports$(), [update.pin.info],    10,
  32, 130+70,  25

    open "LB Ultra Simple Arduino Demo" for dialog as #main
    print #main, "trapclose [quit.main]"
    wait
```

```
[set.pin.settings]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]

    print #main.PinSetting, "selectionindex? index"
    index = index - 1


    Dim pin.setting.options$(100)
    pin.setting.options$(1) = "set"
    pin.setting.options$(2) = "get"
    pin.setting.options$(3) = "pwm.out"
    pin.setting.options$(4) = "pwm.in"
    pin.setting.options$(5) = "servo"


    WindowWidth = 195
    WindowHeight = 120
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)


    combobox #PinSetting.Selection, pin.setting.options$(, [Pin.Settin
g.Done] ,   15,  27, 160, 125
    statictext #PinSetting.statictext2, "Select Pin Setting",  15,   7
, 170,  20
    button #PinSetting.button3,"Done",[Pin.Setting.Done], UL,  15,  57
, 160,  25

    open "Pin Setting" for dialog as #PinSetting
    print #PinSetting, "font ms_sans_serif 10"
    print #PinSetting, "trapclose [Pin.Setting.Done]"
wait



[Pin.Setting.Done]
    print #PinSetting.Selection, "contents? text$"
    ad.pinSetting$(Com.port.Selection, index) = text$
    close #PinSetting
    goto [Update.Pins]




[set.pin.output]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
```

```
    print #main.PinStatuses, "selectionindex? index"
    index = index - 1


    if ad.pinSetting$(Com.port.Selection, index) = "set" then
        'this code will flip the curent pin status from true to false
and vice versa
        if ad.pinstatus(Com.port.Selection, index) = 1 then
            ad.pinstatus(Com.port.Selection, index) = 0
        else
            ad.pinstatus(Com.port.Selection, index) = 1
        end if
    end if

    if ad.pinSetting$(Com.port.Selection, index) = "pwm.out" then
        PROMPT "Value to set and send"; value
        ad.pinstatus(Com.port.Selection, index) = value
    end if

    if ad.pinSetting$(Com.port.Selection, index) = "servo" then
        PROMPT "Value to set and send"; value
        ad.pinstatus(Com.port.Selection, index) = value
    end if
    goto [Update.Pins]



[Set.Pin]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    return.value = AD.Set(Com.port.Selection, Pin.Selection, Val.To.Se
nd)
    print #main.ReturnValue, return.value
    goto [update.pin.info]


[Get.Pin]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    return.value = AD.Get(Com.port.Selection, Pin.Selection)
    print #main.ReturnValue, return.value
    goto [update.pin.info]




[PWM.Out]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
```

```
    return.value = AD.PWM.Out(Com.port.Selection, Pin.Selection, Val.T
o.Send)
    print #main.ReturnValue, return.value
    goto [update.pin.info]



[PWM.In]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    return.value = AD.PWM.In(Com.port.Selection, Pin.Selection)
    print #main.ReturnValue, return.value
    goto [update.pin.info]


[Servo.out]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    return.value =  AD.Servo(Com.port.Selection, Pin.Selection, Val.To
.Send)
    print #main.ReturnValue, return.value
    goto [update.pin.info]




[Update.Pins]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    return.value =  AD.Update.All()
    print #main.ReturnValue, return.value
    goto [update.pin.info]



[quit.main]
    close #main
    bla = AD.close.all.the.com.ports()
    end




[Get.The.Selctions.And.Textboxes.From.Main.Win]
    print #main.ValToSend, "!contents? Val.To.Send";
    print #main.ComPortNo, "contents? Com.port.Selection";
    print #main.PinSelector, "selection? Pin.Selection"
    return
```

```
[update.pin.info]
    gosub [Get.The.Selctions.And.Textboxes.From.Main.Win]
    dim pin.statuses$(13)

    dim pin.Settings$(13)

    for x = 0 to 13
        pin.statuses$(x)   =   str$(ad.pinstatus(Com.port.Selection,x)
)
        pin.Settings$(x)  =    ad.pinSetting$(Com.port.Selection,x)
    next x


    print #main.PinSetting, "reload"
    print #main.PinStatuses, "reload"
wait




'Begin Arduino Code here ------------------------------------------
----------
'These functions can be added to any program to provide arduini power
input and output
'Descriptions for functions are in there comments



'these are the funtions you call to get the arduino to do your bidding
function AD.Set(com.port, PinNo, value)
'Set an Arduino pin high or low,
'Will return 1 one if successful, 0 if not
     AD.Set = val(AD.talk.to.device$(com.port,"set ";PinNo;" ";value))
     ad.pinstatus(com.port,PinNo) = value
     ad.pinSetting$(com.port,PinNo) = "set"
end function
```

```
function AD.Get(com.port, PinNo)
'Will return the current state of the input pin
'1 for high, 0 for low
     AD.Get = val(AD.talk.to.device$(com.port,"get ";PinNo))
     ad.pinstatus(com.port,PinNo) = AD.Get
     ad.pinSetting$(com.port,PinNo) = "get"
end function


function AD.PWM.Out(com.port, PinNo, value)
'Set the PWM output on the select pin to the value
'Will return a 1 for success and 0 for failure.
     AD.PWM.Out = val(AD.talk.to.device$(com.port,"pwm.out ";PinNo;" "
;value))
     ad.pinstatus(com.port,PinNo) = value
     ad.pinSetting$(com.port,PinNo) = "pwm.out"
end function


function AD.PWM.In(com.port, PinNo)
'Get the PWM input from the pin
'Will return a value of 0 to 1023
    AD.PWM.In = val(AD.talk.to.device$(com.port,"pwm.in ";PinNo))
    ad.pinSetting$(com.port,PinNo) = "pwm.in"
end function


function AD.Servo(com.port, PinNo, value)
'Get the PWM input from the pin
'Will return a value of 0 to 1023
    AD.Servo = val(AD.talk.to.device$(com.port,"servo ";PinNo;" ";valu
e))
    ad.pinstatus(com.port,PinNo) = value
    ad.pinSetting$(com.port,PinNo) = "servo"
end function




function AD.Update.All()
    for x = 0 to 10
        for y = 0 to 13
            if ad.pinSetting$(x,y) = "set" then bla = AD.Set(x, y, ad.
pinstatus(x,y))
            if ad.pinSetting$(x,y) = "get" then ad.pinstatus(x,y) = AD
```

```
.Get(x, y)
            if ad.pinSetting$(x,y) = "pwm.out" then bla = AD.PWM.Out(x
, y, ad.pinstatus(x,y))
            if ad.pinSetting$(x,y) = "pwm.in" then ad.pinstatus(x,y) =
 AD.PWM.In(x, y)
            if ad.pinSetting$(x,y) = "servo" then bla = AD.Servo(x, y,
 ad.pinstatus(x,y))
        next y
    next x
end function



'end of function to call to have the arduino do your bidding









Function AD.talk.to.device$(port,msg$)
'Send a Message to the arduino and return the result sent back by the
device
'this function is called by the other functions above
    if msg$ <> "" then
        'Open the com port
        select case port
            case 1
                if ad.com.ports(1) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort1
                    ad.com.ports(1) = 1
                end if
            case 2
                if ad.com.ports(2) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort2
                    ad.com.ports(2) = 1
                end if
            case 3
                if ad.com.ports(3) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
```

```
andom as #ArduinoCOMPort3
                    ad.com.ports(3) = 1
                end if
            case 4
                if ad.com.ports(4) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort4
                    ad.com.ports(4) = 1
                end if
            case 5
                if ad.com.ports(5) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort5
                    ad.com.ports(5) = 1
                end if
            case 6
                if ad.com.ports(6) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort6
                    ad.com.ports(6) = 1
                end if
            case 7
                if ad.com.ports(7) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort7
                    ad.com.ports(7) = 1
                end if
            case 8
                if ad.com.ports(8) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort8
                    ad.com.ports(8) = 1
                end if
            case 9
                if ad.com.ports(9) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort9
                    ad.com.ports(9) = 1
                end if
            case 10
                if ad.com.ports(10) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort10
                    ad.com.ports(10) = 1
                end if
        end select
```

```
'Send message to device

select case port
    case 1
        print #ArduinoCOMPort1, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort1) > 0
                temp$ = input$(#ArduinoCOMPort1, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 2
        print #ArduinoCOMPort2, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort2) > 0
                temp$ = input$(#ArduinoCOMPort2, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 3
        print #ArduinoCOMPort3, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort3) > 0
                temp$ = input$(#ArduinoCOMPort3, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
```

```
                exit while
            end if
        wend
    wend
case 4
    print #ArduinoCOMPort4, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort4) > 0
            temp$ = input$(#ArduinoCOMPort4, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
            end if
        wend
    wend
case 5
    print #ArduinoCOMPort5, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort5) > 0
            temp$ = input$(#ArduinoCOMPort5, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
            end if
        wend
    wend
case 6
    print #ArduinoCOMPort6, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort6) > 0
            temp$ = input$(#ArduinoCOMPort6, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
```

```
            end if
        wend
    wend
case 7
    print #ArduinoCOMPort7, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort7) > 0
            temp$ = input$(#ArduinoCOMPort7, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
            end if
        wend
    wend
case 8
    print #ArduinoCOMPort8, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort8) > 0
            temp$ = input$(#ArduinoCOMPort8, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
            end if
        wend
    wend
case 9
    print #ArduinoCOMPort9, msg$

    'Wait until there is a carriage return
    while foundLF = 0
        while lof(#ArduinoCOMPort9) > 0
            temp$ = input$(#ArduinoCOMPort9, 1)
            buffer$ = buffer$ + temp$

            if temp$ = chr$(13) then
                foundLF = 1
                exit while
            end if
```

```
                wend
            wend
        case 10
            print #ArduinoCOMPort10, msg$

            'Wait until there is a carriage return
            while foundLF = 0
                while lof(#ArduinoCOMPort10) > 0
                    temp$ = input$(#ArduinoCOMPort10, 1)
                    buffer$ = buffer$ + temp$

                    if temp$ = chr$(13) then
                        foundLF = 1
                        exit while
                    end if
                wend
            wend



    end select

    'Place the returned text in to AD.talk.to.device$
    AD.talk.to.device$ = buffer$
else
    Notice "Improper Message Received, Must contain a command"
end if
End function
```

```
function AD.close.all.the.com.ports()
'a function to call on program exit to close all the com ports gracefu
lly
  if ad.com.ports(1) = 1 then
      close #ArduinoCOMPort1
  end if

  if ad.com.ports(2) = 1 then
      close #ArduinoCOMPort2
```

```
  end if

  if ad.com.ports(3) = 1 then
      close #ArduinoCOMPort3
  end if

  if ad.com.ports(4) = 1 then
      close #ArduinoCOMPort4
  end if

  if ad.com.ports(5) = 1 then
      close #ArduinoCOMPort5
  end if

  if ad.com.ports(6) = 1 then
      close #ArduinoCOMPort6
  end if

  if ad.com.ports(7) = 1 then
      close #ArduinoCOMPort7
  end if

  if ad.com.ports(8) = 1 then
      close #ArduinoCOMPort8
  end if

  if ad.com.ports(9) = 1 then
      close #ArduinoCOMPort9
  end if

  if ad.com.ports(10) = 1 then
      close #ArduinoCOMPort10
  end if

end function
```

Just the Library and function descriptions

The bare naked functions are below with commented explanations for what they do.

```
'This must be placed at the beginning of your program
    dim ad.pinstatus(10,100)
```

```
    dim ad.pinSetting$(10,100)

    for x = 0 to 10
        for y = 0 to 100
            ad.pinSetting$(x,y) = "Un-Set"
        next y
    next x

'Put your arduino code here ----------------------------------------
------------------------------------------------------------------
----------------------------------
```

```
'Begin Arduino Code here ------------------------------------------
----------
'These functions can be added to any program to provide arduini power
input and output
'Descriptions for functions are in there comments
```

```
'these are the funtions you call to get the arduino to do your bidding
function AD.Set(com.port, PinNo, value)
'Set an Arduino pin high or low,
'Will return 1 one if successful, 0 if not
     AD.Set = val(AD.talk.to.device$(com.port,"set ";PinNo;" ";value))
     ad.pinstatus(com.port,PinNo) = value
     ad.pinSetting$(com.port,PinNo) = "set"
end function


function AD.Get(com.port, PinNo)
'Will return the current state of the input pin
'1 for high, 0 for low
     AD.Get = val(AD.talk.to.device$(com.port,"get ";PinNo))
```

```
     ad.pinstatus(com.port,PinNo) = AD.Get
     ad.pinSetting$(com.port,PinNo) = "get"
end function


function AD.PWM.Out(com.port, PinNo, value)
'Set the PWM output on the select pin to the value
'Will return a 1 for success and 0 for failure.
     AD.PWM.Out = val(AD.talk.to.device$(com.port,"pwm.out ";PinNo;" "
;value))
     ad.pinstatus(com.port,PinNo) = value
     ad.pinSetting$(com.port,PinNo) = "pwm.out"
end function


function AD.PWM.In(com.port, PinNo)
'Get the PWM input from the pin
'Will return a value of 0 to 1023
    AD.PWM.In = val(AD.talk.to.device$(com.port,"pwm.in ";PinNo))
    ad.pinSetting$(com.port,PinNo) = "pwm.in"
end function


function AD.Servo(com.port, PinNo, value)
'Get the PWM input from the pin
'Will return a value of 1 if successful
    AD.Servo = val(AD.talk.to.device$(com.port,"servo ";PinNo;" ";valu
e))
    ad.pinstatus(com.port,PinNo) = value
    ad.pinSetting$(com.port,PinNo) = "servo"
end function




function AD.Update.All()
    for x = 0 to 10
        for y = 0 to 13
            if ad.pinSetting$(x,y) = "set" then bla = AD.Set(x, y, ad.
pinstatus(x,y))
            if ad.pinSetting$(x,y) = "get" then ad.pinstatus(x,y) = AD
.Get(x, y)
            if ad.pinSetting$(x,y) = "pwm.out" then bla = AD.PWM.Out(x
, y, ad.pinstatus(x,y))
            if ad.pinSetting$(x,y) = "pwm.in" then ad.pinstatus(x,y) =
 AD.PWM.In(x, y)
```

```
            if ad.pinSetting$(x,y) = "servo" then bla = AD.Servo(x, y,
 ad.pinstatus(x,y))
        next y
    next x
end function



'end of function to call to have the arduino do your bidding









Function AD.talk.to.device$(port,msg$)
'Send a Message to the arduino and return the result sent back by the
device
'this function is called by the other functions above
    if msg$ <> "" then
        'Open the com port
        select case port
            case 1
                if ad.com.ports(1) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort1
                    ad.com.ports(1) = 1
                end if
            case 2
                if ad.com.ports(2) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort2
                    ad.com.ports(2) = 1
                end if
            case 3
                if ad.com.ports(3) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort3
                    ad.com.ports(3) = 1
                end if
            case 4
                if ad.com.ports(4) <> 1 then
```

```
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort4
                    ad.com.ports(4) = 1
                end if
            case 5
                if ad.com.ports(5) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort5
                    ad.com.ports(5) = 1
                end if
            case 6
                if ad.com.ports(6) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort6
                    ad.com.ports(6) = 1
                end if
            case 7
                if ad.com.ports(7) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort7
                    ad.com.ports(7) = 1
                end if
            case 8
                if ad.com.ports(8) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort8
                    ad.com.ports(8) = 1
                end if
            case 9
                if ad.com.ports(9) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort9
                    ad.com.ports(9) = 1
                end if
            case 10
                if ad.com.ports(10) <> 1 then
                    open "COM" ; port ; ":9600,n,8,1,ds0,cs0,rs" for r
andom as #ArduinoCOMPort10
                    ad.com.ports(10) = 1
                end if
        end select


        'Send message to device

        select case port
```

```
    case 1
        print #ArduinoCOMPort1, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort1) > 0
                temp$ = input$(#ArduinoCOMPort1, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 2
        print #ArduinoCOMPort2, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort2) > 0
                temp$ = input$(#ArduinoCOMPort2, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 3
        print #ArduinoCOMPort3, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort3) > 0
                temp$ = input$(#ArduinoCOMPort3, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 4
```

```
        print #ArduinoCOMPort4, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort4) > 0
                temp$ = input$(#ArduinoCOMPort4, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 5
        print #ArduinoCOMPort5, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort5) > 0
                temp$ = input$(#ArduinoCOMPort5, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 6
        print #ArduinoCOMPort6, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort6) > 0
                temp$ = input$(#ArduinoCOMPort6, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 7
        print #ArduinoCOMPort7, msg$
```

```
        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort7) > 0
                temp$ = input$(#ArduinoCOMPort7, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 8
        print #ArduinoCOMPort8, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort8) > 0
                temp$ = input$(#ArduinoCOMPort8, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 9
        print #ArduinoCOMPort9, msg$

        'Wait until there is a carriage return
        while foundLF = 0
            while lof(#ArduinoCOMPort9) > 0
                temp$ = input$(#ArduinoCOMPort9, 1)
                buffer$ = buffer$ + temp$

                if temp$ = chr$(13) then
                    foundLF = 1
                    exit while
                end if
            wend
        wend
    case 10
        print #ArduinoCOMPort10, msg$
```

```
                'Wait until there is a carriage return
                while foundLF = 0
                    while lof(#ArduinoCOMPort10) > 0
                        temp$ = input$(#ArduinoCOMPort10, 1)
                        buffer$ = buffer$ + temp$

                        if temp$ = chr$(13) then
                            foundLF = 1
                            exit while
                        end if
                    wend
                wend



        end select

        'Place the returned text in to AD.talk.to.device$
        AD.talk.to.device$ = buffer$
    else
        Notice "Improper Message Received, Must contain a command"
    end if
End function
```

```
function AD.close.all.the.com.ports()
'a function to call on program exit to close all the com ports gracefu
lly
  if ad.com.ports(1) = 1 then
      close #ArduinoCOMPort1
  end if

  if ad.com.ports(2) = 1 then
      close #ArduinoCOMPort2
  end if

  if ad.com.ports(3) = 1 then
      close #ArduinoCOMPort3
  end if
```

```
  if ad.com.ports(4) = 1 then
      close #ArduinoCOMPort4
  end if

  if ad.com.ports(5) = 1 then
      close #ArduinoCOMPort5
  end if

  if ad.com.ports(6) = 1 then
      close #ArduinoCOMPort6
  end if

  if ad.com.ports(7) = 1 then
      close #ArduinoCOMPort7
  end if

  if ad.com.ports(8) = 1 then
      close #ArduinoCOMPort8
  end if

  if ad.com.ports(9) = 1 then
      close #ArduinoCOMPort9
  end if

  if ad.com.ports(10) = 1 then
      close #ArduinoCOMPort10
  end if

end function
```

Credits and thank yous

With out the folks listed below the foundation of this project would not exist and serial communications would still be a mystery to me.
I appreciate the time an effort and frustration these people had to deal with getting this project going.

Chris Iverson
Colin McMurchie
Rod
metro

Table of Contents