

infraLED Zeppelin

Nick Wagner, Austin Jurgensmeyer, and Christopher Record

12/16/08



The infraLED Zeppelin is a blimp capable of autonomous and remote control flight. In either mode, the blimp uses the pulse width output of an ultrasonic range finder along with a proportional control algorithm to decide on a servo position that will maintain a set altitude during flight.

In RC mode, a transmitter sends serial control codes to the blimp which are interpreted and enacted. The transmitter displays the commands available on a 20x4 character lcd display. These commands consist of forward, reverse, left turn, right turn, take off, and land and are selected via the use of a 12 key keypad.

In autonomous mode, the blimp makes its own directional decisions by use of an infrared beacon and 4 infrared phototransistors. The blimp employs a signal to noise ratio algorithm to make these decisions.

The control system of the blimp consists of 4 different elements: a serial buffer, altitude controller, directional controller, and the main controller. The serial buffer receives rf serial commands and stores them until the main controller is ready. It then employs a software based handshaking. Once the command is sent from the serial buffer to the main controller, the buffer is ready to receive the next command.

The altitude controller takes input from the ultrasonic range finder and then processes it using the following system of equations:

*if (alt > 500) & (alt < 900) then pulse = 375 - .35 * alt*

if (alt > 900) then pulse = 60

if (alt < 500) then pulse = 200

where 'alt' is the raw altitude from the sensor and pulse is the pulse width sent to the servo to set the appropriate correction angle to the thrust bar. The altitude controller also communicates to the main controller over two wires. Both wires use simple binary communication and correspond to taking off and landing states. When either wire is high, the altitude controller sets the thrust bar to either vertical up or vertical down states corresponding to the respective wire.

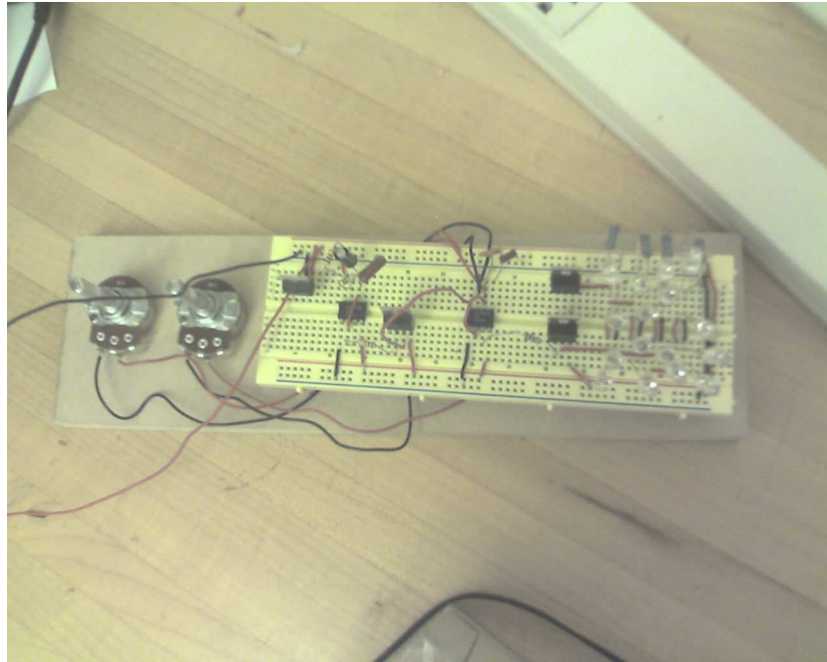
The directional controller takes input from 4 infrared phototransistors and outputs 4 logic level states to the main controller. When the infrared phototransistor receives a pulse from the infrared beacon, the output goes low. The directional controller counts low pulses from each of the 4 phototransistors and compares them to see which has the larger number of pulses. The controller then outputs the results of this comparison by making one of the four wires high. Each of the four wires corresponds to a direction, north, south, east, and west. When in autonomous mode, the main controller reads each of those four wires to see which direction it needs to travel and sets the motor states expected.

The main controller is the link between all of the specialization controllers. It first checks the mode switch to see if the blimp is in autonomous or RC mode. When the serial buffer sends information to the main controller, such as takeoff or landing, it communicates those actions to the altitude controller. When the serial buffer sends directional commands, the main controller interprets them and sets the correct logic levels for the motor controller. In autonomous mode, the main controller interprets information sent by the directional controller and sets the correct logic levels for the motor controller.



The menu driven display of the transmitter lets the user choose which command set is active. The transmitter is primarily used in RC mode. In that menu, the user is capable of going forwards, backwards, left, right, take off and land. In auto mode, the user can choose to take off or land. The commands are sent serially through an RF transmitter and received by the serial buffer on the blimp. Other features on the transmitter include power and data transfer led's, a speaker which plays a nice startup tune and beeps on error, an LCD for communicating data, and a keypad for inputting data.

The IR phototransistors selected for the sensor array on the blimp contained noise suppression which our original beacon design was not able to avoid, thus our new design came about. In the current design of the beacon, two 555 timer circuits are used to modulate the IR LED array. The first of which modulates the signal necessary to the IR phototransistors at 56 kHz. The second 555 timer modulates the power to the first 555 timer to get around the noise suppression. In the current design, when a pulse from the beacon is received, the level on the phototransistor goes low. It is these pulses that are counted for directional decisions.



Appendices

- A. Transmitter PIC Basic Code
- B. Serial Buffer PIC Basic Code
- C. Altitude Controller PIC Basic Code
- D. Directional Controller PIC Basic Code
- E. Main Blimp PIC Basic Code

Appendix A – Transmitter PIC Basic Pro Code

```
*****
'* Name      : BlimpTX.BAS                               *
'* Author    : AustinJ, ChrisR, NickW                   *
'* Notice    : Copyright (c) 2008                       *
'*           : All Rights Reserved                      *
'* Date      : 10/9/2008                                 *
'* Version   : 1.0                                       *
'* Notes     :                                           *
'*           :                                           *
*****

'Tell the pic where to find the LCD
Define LCD_DREG      PORTD
Define LCD_DBIT      4
Define LCD_RSREG     PORTE
Define LCD_RSBIT     0
Define LCD_EREG      PORTE
Define LCD_EBIT      1

'Using a 4 line LCD
DEFINE LCD_LINES 4

'Variable names to keep track of info.
key      VAR   Byte
alt_tens VAR   Byte
alt_ones VAR   Byte
alt      VAR   Byte
counter  var   byte
airborn  VAR   bit
dontexit VAR   bit

' Lets give some pins some names!
col1 var PORTB.0
col2 var PORTB.1
col3 var PORTB.2
row1 var PORTB.3
row2 var PORTB.4
row3 var PORTB.5
row4 var PORTB.6
tx   var PORTD.0
txled var PORTD.1

ADCON0 = 7      'Turn off A to D
ADCON1 = 7      'Turn off A to D

TRISB = %11111000 'Set PORTB outputs and inputs
TRISD = %00000000 'Set PORTD outputs
TRISE = $00       'Set PORTE outputs
OPTION_REG = $7f

SEROUT2 tx, 1646, ["D"] 'Send dummy bit to make serout2 work.
```

```
HIGH col1 : HIGH col2 : HIGH col3
```

```
pause 500
```

```
PrintIntro:
```

```
  lcdout $fe, 1, "InfraLED Zeppelin"
```

```
  lcdout $fe, $c0, "Austin J."
```

```
  lcdout $fe, $94, "Chris R."
```

```
  lcdout $fe, $d4, "Nick W."
```

```
  Pause 500
```

```
  Sound PortD.2, [99, 50, 90, 25, 90, 25, 94, 50, 90, 50, 0, 25, 98,  
50, 99, 50]
```

```
  pause 1000
```

```
presssomething:
```

```
  lcdout $fe, 1, "1) Goto RC Mode"
```

```
  lcdout $fe, $c0, "2) Goto Auto Mode"
```

```
  key = 12
```

```
  while key==12
```

```
    GOSUB getkey
```

```
    if (key > 2) & (Key <> 12) | (key == 0) then gosub
```

```
dosound
```

```
  wend
```

```
  if key == 1 Then rcmode
```

```
  if key == 2 then goauto
```

```
goto presssomething
```

```
END
```

```
goauto:
```

```
  pause 1000
```

```
  lcdout $fe, 1, "1) Takeoff"
```

```
  lcdout $fe, $c0, "2) Land"
```

```
  lcdout $fe, $94, "3) Exit"
```

```
  key=12
```

```
  while key==12
```

```
    GOSUB getkey
```

```
    if (key > 3) & (key <> 12) | (key == 0) Then
```

```
      gosub dosound
```

```
    EndIf
```

```
  wend
```

```
  if key == 1 Then gosub takeoff
```

```
  if key == 2 then gosub land
```

```
  if key == 3 then printintro
```

```
goto goauto
```

```
rcmode:
```

```
  dontexit = 1
```

```
  pause 1000
```

```

rcloop:

    lcdout $fe, 1, "    2) Forward"
    lcdout $fe, $c0, " 4) Left  6) Right"
    lcdout $fe, $94, "    5) Back"
    lcdout $fe, $d4, "7)TkOff 8)Lnd #)Exit"

    key = 12
    while key==12
        GOSUB getkey
    wend

    if key == 2 THEN gosub forward
    if key == 4 then gosub left
    if key == 5 then gosub back
    if key == 6 then gosub right
    if key == 7 then gosub takeoff
    if key == 8 then gosub land
    if (key == 1) | (key == 3) | (key == 0) | (key > 8) & (key <>
11) THEN gosub dosound
    if key == 11 then dontexit = 0
    IF dontexit = 1 THEN rclloop

    key = 12
goto printintro

forward:
    SOUND PORTD.2,[50,10,100,10]
    for counter = 1 to 10
        pause 10
        HIGH txled
        SEROUT2 tx, 1646, ["&for*"]
        LOW txled
    next counter
return

left:
    SOUND PORTD.2,[50,10,100,10]
    for counter = 1 to 10
        pause 10
        HIGH txled
        SEROUT2 tx, 1646, ["&lef*"]
        LOW txled
    next counter
return

back:
    SOUND PORTD.2,[50,10,100,10]
    for counter = 1 to 10
        pause 10
        HIGH txled
        SEROUT2 tx, 1646, ["&bck*"]
        LOW txled
    next counter

```

```
return
```

```
right:
```

```
    SOUND PORTD.2,[50,10,100,10]  
    for counter = 1 to 10  
    pause 10  
    HIGH txled  
    SEROUT2 tx, 1646, ["&rig*"]  
    LOW txled  
    next counter
```

```
return
```

```
land:
```

```
    SOUND PORTD.2,[50,10,100,10]  
    for counter = 1 to 10  
    pause 10  
    HIGH txled  
    SEROUT2 tx, 1646, ["&lan*"]  
    LOW txled  
    next counter
```

```
return
```

```
takeoff:
```

```
    SOUND PORTD.2,[50,10,100,10]  
    for counter = 1 to 10  
    PAUSE 10  
    HIGH txled  
    SEROUT2 tx, 1646, ["Atak*"]  
    LOW txled  
    next counter
```

```
return
```

```
dosound:
```

```
    PAUSE 10  
    SOUND PORTD.2,[100,10,50,10]  
    Key = 12
```

```
return
```

```
getkey:
```

```
    Low col1 : High col2 : High col3
```

```
    IF (row1 == 0) Then  
        key = 1  
        return
```

```
    Endif
```

```
    IF (row2 == 0) Then  
        key = 4  
        return
```

```
    Endif
```

```
    IF (row3 == 0) Then  
        key = 7
```

```
        return
    Endif

    IF (row4 == 0) Then
        key = 10
        return
    Endif

    High col1 : Low col2 : High col3

    IF (row1 == 0) Then
        key = 2
        return
    Endif

    IF (row2 == 0) Then
        key = 5
        return
    Endif

    IF (row3 == 0) Then
        key = 8
        return
    Endif

    IF (row4 == 0) Then
        key = 0
        return
    Endif

    High col1 : High col2 : low col3

    IF (row1 == 0) Then
        key = 3
        return
    Endif

    IF (row2 == 0) Then
        key = 6
        return
    Endif

    IF (row3 == 0) Then
        key = 9
        return
    Endif

    IF (row4 == 0) Then
        key = 11
        return
    Endif

return
```

Appendix B – Serial Buffer PIC Basic Code

```
*****
'* Name      : BLIMPInterp.BAS          *
'* Author    : AustinJ, ChrisR, NickW  *
'* Notice    : Copyright (c) 2008      *
'*           : All Rights Reserved     *
'* Date      : 11/19/2008              *
'* Version   : 1.0                     *
'* Notes     : The following code turns *
'*           : the PIC into a serial    *
'*           : buffer for the wireless  *
'*           : RF receiver.            *
'*           :                          *
*****

DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1

ansel = 0 'Turns off analog to digital conversion. Refer to
          'Threaded Design Example A.4 p.296-299 of the textbook
          'for an example of how to configure and use A/D
conversion

Choice VAR Byte[3] 'Store each command in a 3 byte array.

rx VAR PORTB.2      'Input
tx var   PORTB.6    'Output
CTS VAR PORTB.5    'Input
RTS VAR PORTB.4    'Output

Output TX
Input rx
input cts
output rts

serout2 tx, 1648, ["D"] 'Send a dummy variable to make serout work
right.

LOW RTS 'Make sure RTS is low so that MainBlimp doesn't time out.

mainloop:

SERIN2 RX, 1646, [WAIT("&"),STR Choice\3\"*"] 'Wait until we receive
something
                                           'familiar from the RF
receiver.
'Handshaking over CTS and RTS. This makes sure we send what we need to
'MainBlimp

HIGH RTS 'Set RTS high to inform MainBlimp that there is data in the
buffer.
```

```
WHILE(CTS == 0) 'Wait until MainBlimp is ready to receive the data.
Pause 1
WEND

Pause 10 'Pause a bit to make sure SERIN2 is ready.
SEROUT2 tx, 1646, ["&",CHOICE(0),Choice(1),Choice(2),"*"] 'Regurgitate

LOW RTS 'Clear RTS
goto mainloop
```

Appendix C – Altitude Control PIC Basic Code

```
'*****
'* Name      : ALTITUDE                                     *
'* Author    : [select VIEW...EDITOR OPTIONS]             *
'* Notice    : Copyright (c) 2008 [select VIEW...EDITOR OPTIONS] *
'*           : All Rights Reserved                         *
'* Date      : 11/19/2008                                   *
'* Version   : 1.0                                         *
'* Notes     :                                             *
'*           :                                             *
'*****
```

```
ultrasonic var PortB.5
take_off var PortB.7
land VAR PortB.6
servo var PortB.0
```

```
PORTB = 0
```

```
TRISB = %11100000
TRISA = %11111111      ' Set PORTA to all input
Pause 500              ' Wait .5 second
```

```
'Servo maximum and minimum pulse lengths
servomin var byte
servomax var byte
pulse var word
waittime var byte
```

```
'Altitude control variables
setalt var word
inval var word
avg var word
```

```
setalt = 700
servomax = 200
servomin = 60
waittime = 2
```

```
startloop:
IF (land == 1) then doland
IF (take_off == 1) then takeoff
goto startloop
end
```

```
takeoff:
pulse = servomax
pulsout servo, pulse
pause waittime
pulsout servo, pulse
pause waittime
pulsout servo, pulse
```

```

pause waittime
pulsout servo, pulse
pause waittime

pulse = servomin + (servomax + servomin)/2

Goto maintain_alt
end

maintain_alt:
gosub getalt
if (take_off == 0) then startloop
pulsout servo, pulse
goto maintain_alt

getalt:
avg = 0

pulsin ultrasonic, 1, inval
avg = avg + inval
pulsin ultrasonic, 1, inval
avg = avg + inval
avg = avg/2

pulse = 21750-35*avg
pulse = pulse/100

if avg > 450 then pulse = 60
if avg < 50 then pulse = 200

RETURN

goup:
if(pulse > servomin) THEN
pulse = pulse - 5
endif
return

godown:
if(pulse < servomax) THEN
pulse = pulse + 5
endif
return

doland:
pulse = servomin
pulsout servo, pulse
pause waittime
pulsout servo, pulse
pause waittime

```

```
pulsout servo, pulse  
pause waittime  
pulsout servo, pulse  
pause waittime  
goto startloop
```

```
end
```

Appendix D –Directional Control PIC Basic Code

```
*****
'* Name      : DirectionalControl          *
'* Author    : AustinJ, ChrisR, NickW     *
'* Notice    : Copyright (c) 2008         *
'*           : All Rights Reserved        *
'* Date      : 11/19/2008                 *
'* Version   : 2.2                        *
'* Notes     :                             *
*****
DEFINE OSC 8
OSCCON.4 = 1 'Sets the internal oscillator frequency to 8 MHz
OSCCON.5 = 1
OSCCON.6 = 1

ansel = 0 'Turns off analog to digital conversion. Refer to
          'Threaded Design Example A.4 p.296-299 of the textbook
          'for an example of how to configure and use A/D conversion

IRN var Portb.0
IRS var Portb.1
IRE var Portb.2
IRW var Portb.3
b0 VAR Portb.4
b1 VAR Portb.5
b2 var Portb.6
b3 VAR Portb.7

COUNTS var word[4]
counter var word
result var bit[4]
result3 var byte
decreresult var byte

TRISB = %00001111
portb = 0

loop:

COUNTS[0] = 0
COUNTS[1] = 0
COUNTS[2] = 0
COUNTS[3] = 0

for counter = 1 to 9000
if (IRN = 0) then COUNTS[0] = COUNTS[0] + 1
if (IRS = 0) then COUNTS[1] = COUNTS[1] + 1
if (IRE = 0) then COUNTS[2] = COUNTS[2] + 1
if (IRW = 0) then COUNTS[3] = COUNTS[3] + 1
pauseus 10
next counter

IF (counts[0] > counts[2]) THEN
```

```
result3 = 0
ELSE
result3 = 2
endif

IF (counts[1] > counts[3]) THEN
result3 = 1
ELSE
result3 = 3
endif

if result3 = 0 then
result[0] = 0
result[1] = 0
result[2] = 0
result[3] = 0
endif

if result3 = 1 then
result[0] = 0
result[1] = 1
result[2] = 0
result[3] = 0
endif

if result3 = 2 then
result[0] = 0
result[1] = 0
result[2] = 1
result[3] = 0
endif

if result3 = 3 then
result[0] = 0
result[1] = 0
result[2] = 0
result[3] = 1
endif

b0 = result[0]
b1 = result[1]
b2 = result[2]
b3 = result[3]

pause 500

goto loop
```

Appendix E –Directional Control PIC Basic Code

```

'*****
'* Name      : MainBlimp.BAS                                     *
'* Author    : [select VIEW...EDITOR OPTIONS]                   *
'* Notice    : Copyright (c) 2008 [select VIEW...EDITOR OPTIONS] *
'*           : All Rights Reserved                               *
'* Date      : 11/19/2008                                        *
'* Version   : 1.0                                              *
'* Notes     :                                                  *
'*           :                                                  *
'*****
ADCON0 = 7          'Turn off A to D
ADCON1 = 7          'Turn off A to D

' Define PINS
rxdata    var      PortB.0
rxCTS     var      PortB.1
rxRTS     var      PortB.2

IRN       VAR      PortB.3
IRS       VAR      PortB.4
IRE       VAR      PortB.5
IRW       VAR      PortB.6

mode      var      PortB.7

takeoff   VAR      PortC.0
land      var      PortC.1

thrustAA  var      PortD.0
thrustAB  VAR      PortD.1
thrustBA  VAR      PortD.6
thrustBB  VAR      PortD.7
direction var byte

Choice     VAR Byte[3] 'RX Data Storage

TRISB = %11111101
TRISC = %00000000
TRISD = %00000000

LOW thrustAA :Low thrustBB :LOW thrustab :low thrustba :low rxcts
high takeoff :Low land

IF mode = 0 then rcmode

automode:

IF (rxRTS = 1) THEN gosub rxinterpdata
IF ((Choice[0] = "t")&&(Choice[1] = "a")&&(Choice[2] = "k")) THEN
high takeoff
LOW LAND
GOSUB goforward

```

```

pause 3000
endif

IF ((Choice[0] = "l")&&(Choice[1] = "a")&&(Choice[2] = "n")) THEN
Low takeoff
pause 5
high land
gosub goforward
endif

'if (takeoff = 1) THEN
if (IRN = 1) then GOSUB goforward
IF (IRS = 1) then GOSUB autoback
IF (IRE = 1) THEN GOSUB goright
IF (IRW = 1) THEN GOSUB goleft
pause 1000
'ENDIF
goto automode

rcmode:

IF (rxRTS = 1) THEN
gosub rxinterpdata

IF ((Choice[0] = "t")&&(Choice[1] = "a")&&(Choice[2] = "k")) THEN
LOW Land
pause 5
high takeoff
GOSUB goforward
choice = 0
pause 3000
endif

IF ((Choice[0] = "l")&&(Choice[1] = "a")&&(Choice[2] = "n")) THEN
Low takeoff
pause 5
high land
gosub goforward
choice = 0
endif

IF ((Choice[0] = "f")&&(Choice[1] = "o")&&(Choice[2] = "r")) THEN gosub
goforward
IF ((Choice[0] = "b")&&(Choice[1] = "c")&&(Choice[2] = "k")) THEN gosub
gobackward
IF ((Choice[0] = "l")&&(Choice[1] = "e")&&(Choice[2] = "f")) THEN gosub
goleft
IF ((Choice[0] = "r")&&(Choice[1] = "i")&&(Choice[2] = "g")) THEN gosub
goright

ENDIF
goto rcmode

'Both motors forwards

```

```
goforward:
if (direction <> "N") then
LOW thrustAA
LOW thrustab
low thrustba
Low thrustBB
PAUSE 500
ENDIF
```

```
high THRUSTAA
LOW THRUSTAB
HIGH THRUSTBA
LOW THRUSTBB
choice = 0
direction = "N"
return
```

```
'Both motors backwards
gobackward:
if (direction <> "S") then
LOW thrustAA
LOW thrustab
low thrustba
Low thrustBB
PAUSE 500
ENDIF
LOW THRUSTAA
HIGH THRUSTAB
LOW THRUSTBA
HIGH THRUSTBB
choice = 0
direction = "S"
return
```

```
'Right motor forward, left motor back
goright:
if (direction <> "E") then
LOW thrustAA
LOW thrustab
low thrustba
Low thrustBB
PAUSE 500
ENDIF
LOW THRUSTAA
HIGH THRUSTAB
HIGH THRUSTBA
LOW THRUSTBB
Choice = 0
DIRECTION = "E"
return
```

```
'Left motor back, right motor forward
goleft:
if (direction <> "W") then
```

```
LOW thrustAA
LOW thrustab
low thrustba
Low thrustBB
PAUSE 500
ENDIF
high THRUSTAA
LOW THRUSTAB
LOW THRUSTBA
HIGH THRUSTBB
Choice = 0
DIRECTION = "W"
return
```

```
autoback:
gosub goleft
return
```

```
rxinterpdata:
High rxCTS
pause 5
SERIN2 rxDATA, 1646, [WAIT("&"),STR Choice\3\"*"]
LOW rxCTS
RETURN
```

