Automatic Animal Feeder Abstract

Project Registration No: WZ1175

Prepared for:
WIZnet 2014 Design Challenge

The Automatic Animal Feeder system is designed to automatically feed hay to small farm animals such as goats and sheep. The system provides owners of these animals the freedom to be away from their animals for up to a week at a time and to be confident that their animals are being fed properly. The control system for the feeder uses a Microchip ChipKit Max32 processor module, and the network interface connection is provided by the WIZnet WIZ550io network module This design extends the Internet of Things to the barnyard which allows the operation of the feeder to be controlled and monitored remotely via the Internet.

The Automatic Animal Feeder consists of the chipKit Max32 development board, along with two extension boards that host the unique interfaces required by the system. The Control Board hosts the chipKit processor, WIZ550io module, power supply, system interfaces, and a host of housekeeping and monitoring functions. The AC Control Board contains the solid state relays that control the feeder and several auxiliary outputs.   In addition, this board also hosts a watchdog monitor to ensure the overall system operates safely. A block diagram is shown in Figure 1 and the completed project is shown in Figure 2. A photo of the control electronics is shown in Figure 3.

The feeder consists of 6 vertical feed bins to hold individual pads of hay. Each pad of hay, which is approximately 24” x 15” x 4”, is sufficient to feed the animals for one day. The bottom of the vertical bin consists of a trap door which holds the hay in the bin. At the appropriate time, either based on time of day or via command received from the Internet, the control system commands a linear actuator which triggers the release mechanism, which causes the trap door to open. The pad of hay then drops into the area below the feeder, which is accessible to the animals. A sensor is attached to each trap door so the controller can verify that door opened correctly. If the controller senses that a problem existed that prevented the door from opening, it will automatically select a different bin to release. The web interface for monitoring and controlling the system operation is shown in Figure 4.

A short two minute video that demonstrates the operation of the system is provided in the file “Demo_video.wmv” located in the Demo_video directory.

The software design for the Automatic Animal Feeder utilizes custom software written in the C language using the Microchip MPLAB X development environment to implement all the functions required to control the feeder. This custom software is integrated with the Microchip TCPIP stack, which provides the web server interface and all the other standard network functions. At the time this project was completed, interface drivers between the Microchip TCPIP stack and the WIZ550io module were not available so a significant part of this project was to develop the interface between the WIZ550io module and the Microchip stack.

The linear actuators, which are part of the mechanical release mechanism, have the unfortunate characteristic that they are not rated for continuous operation. Under normal operating conditions, this is not an issue since each actuator is energized for only 1-second at a time. However, since the actuators are under software control there was a possibility that a software failure could result in an actuator being energized continuously potentially resulting in a fire hazard. This condition violates one of my fundamental design rules, which is “A single software failure must never result in damage to life or property”. To mitigate this risk, a watchdog function was implemented to continuously monitor the actuators and to disable them in the event they are commanded on too long.

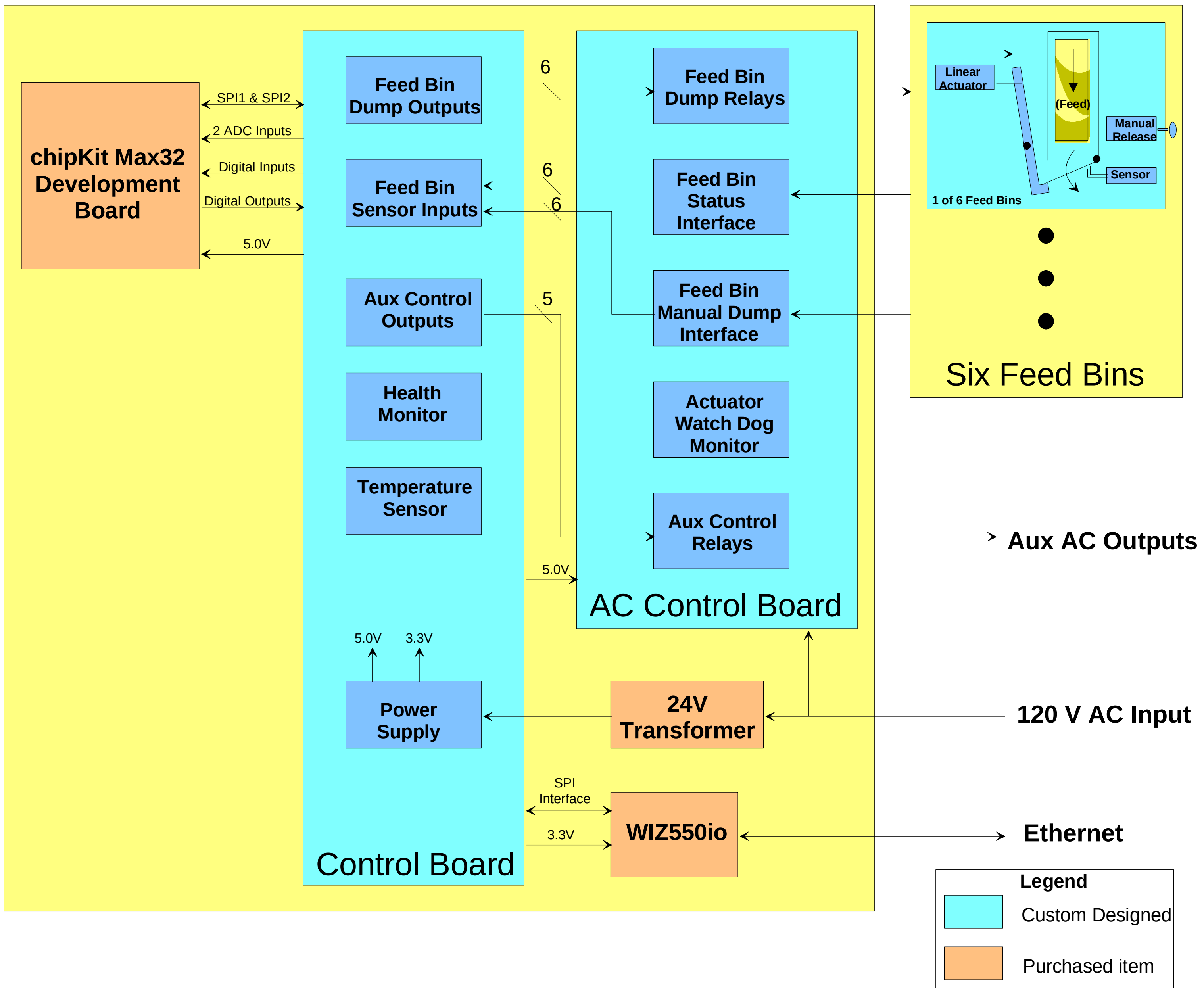


Figure 1 Automatic Animal Feeder Block Diagram – The feeder system consists of the chipKit Max32 controller, the WIZ550io network interface, and two custom boards. In addition, mechanical mechanisms are included to physically control the release of the animal feed.

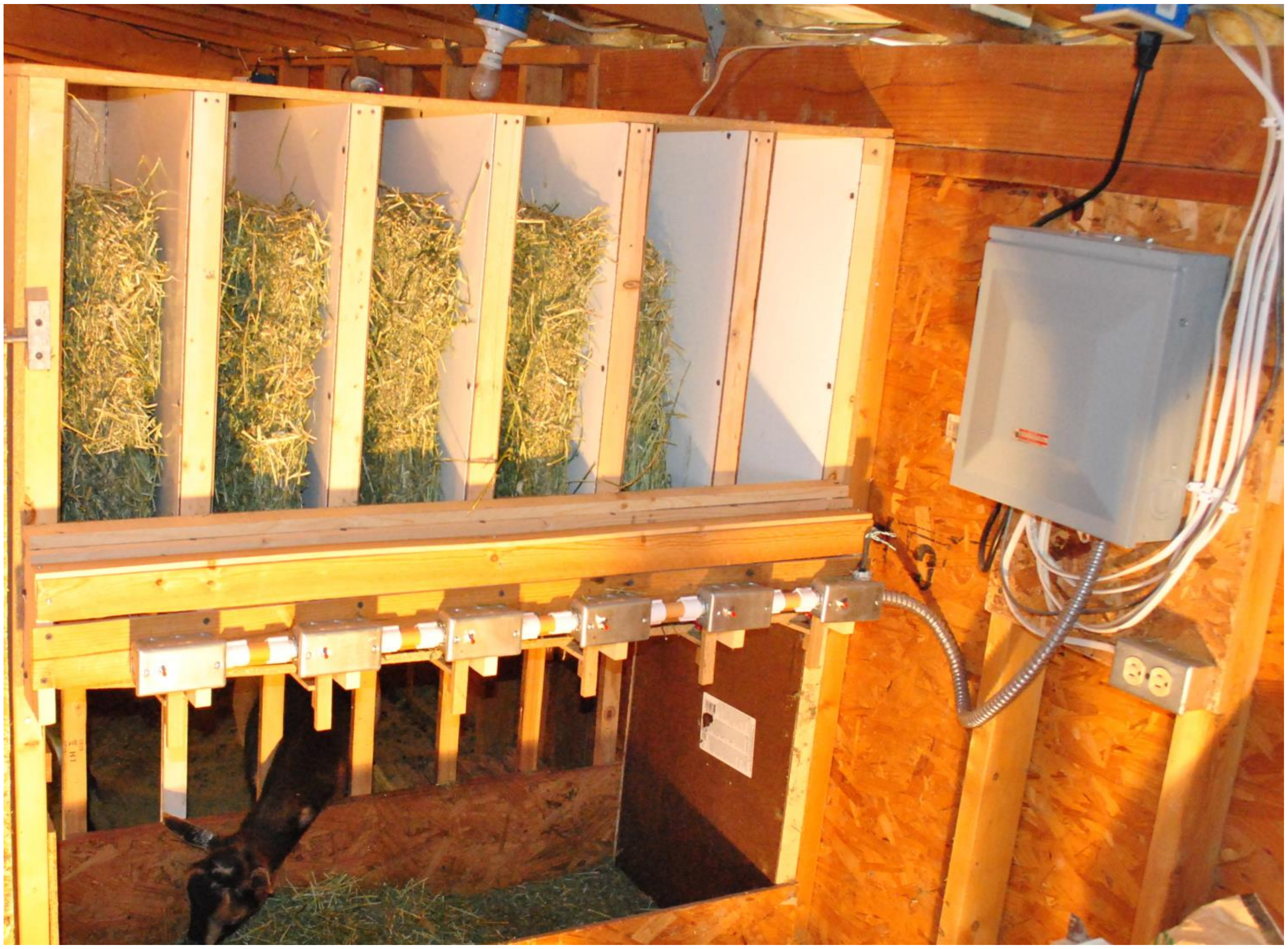


Figure 2 Photo of Completed Automatic Animal Feeder System – The six feed bins are shown in the upper left portion of the photo and the gray electrical box on the right contains all the control electronics to control the feeder. The electronics also contains a web server to provide remote control and monitoring via the Internet.

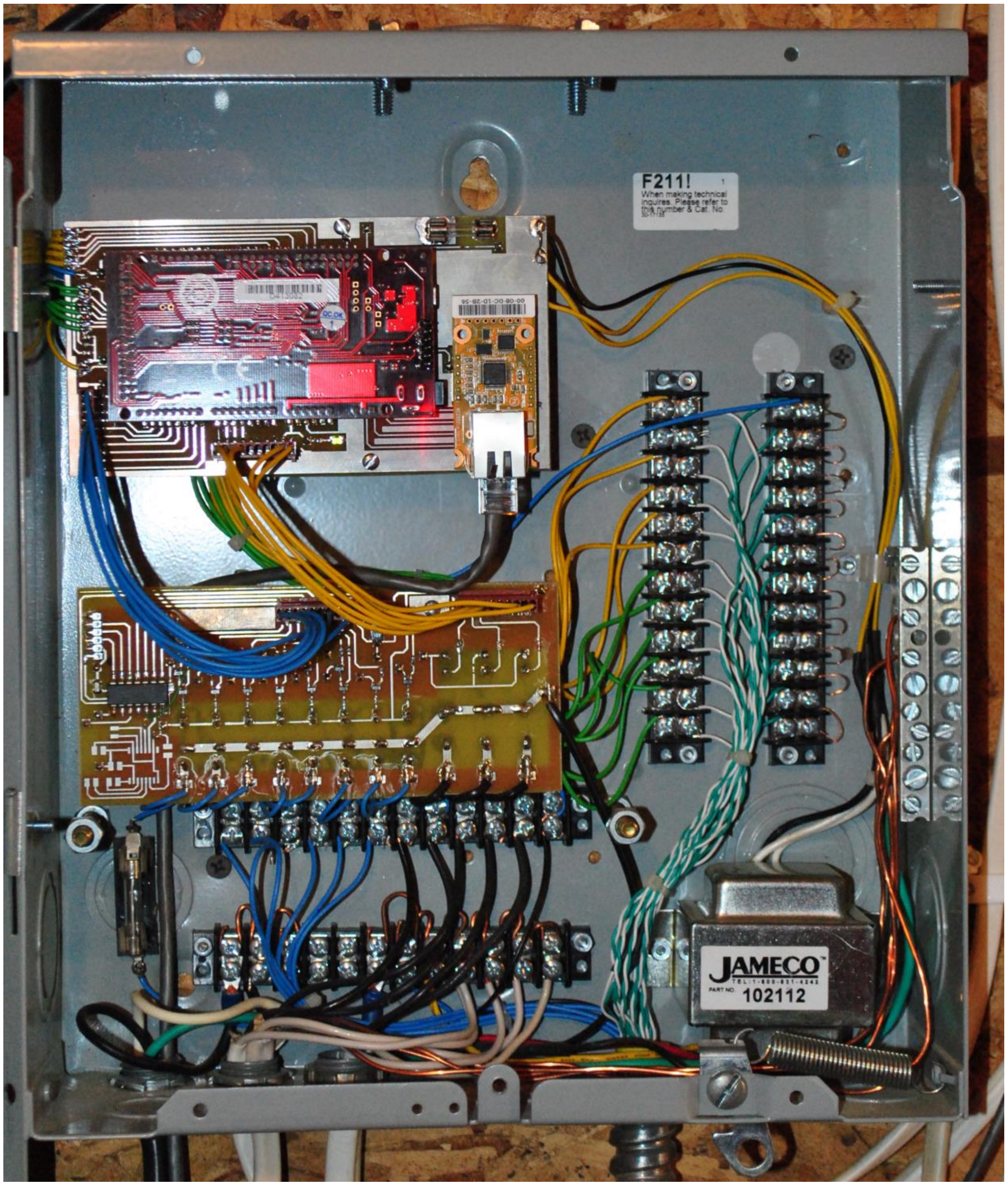
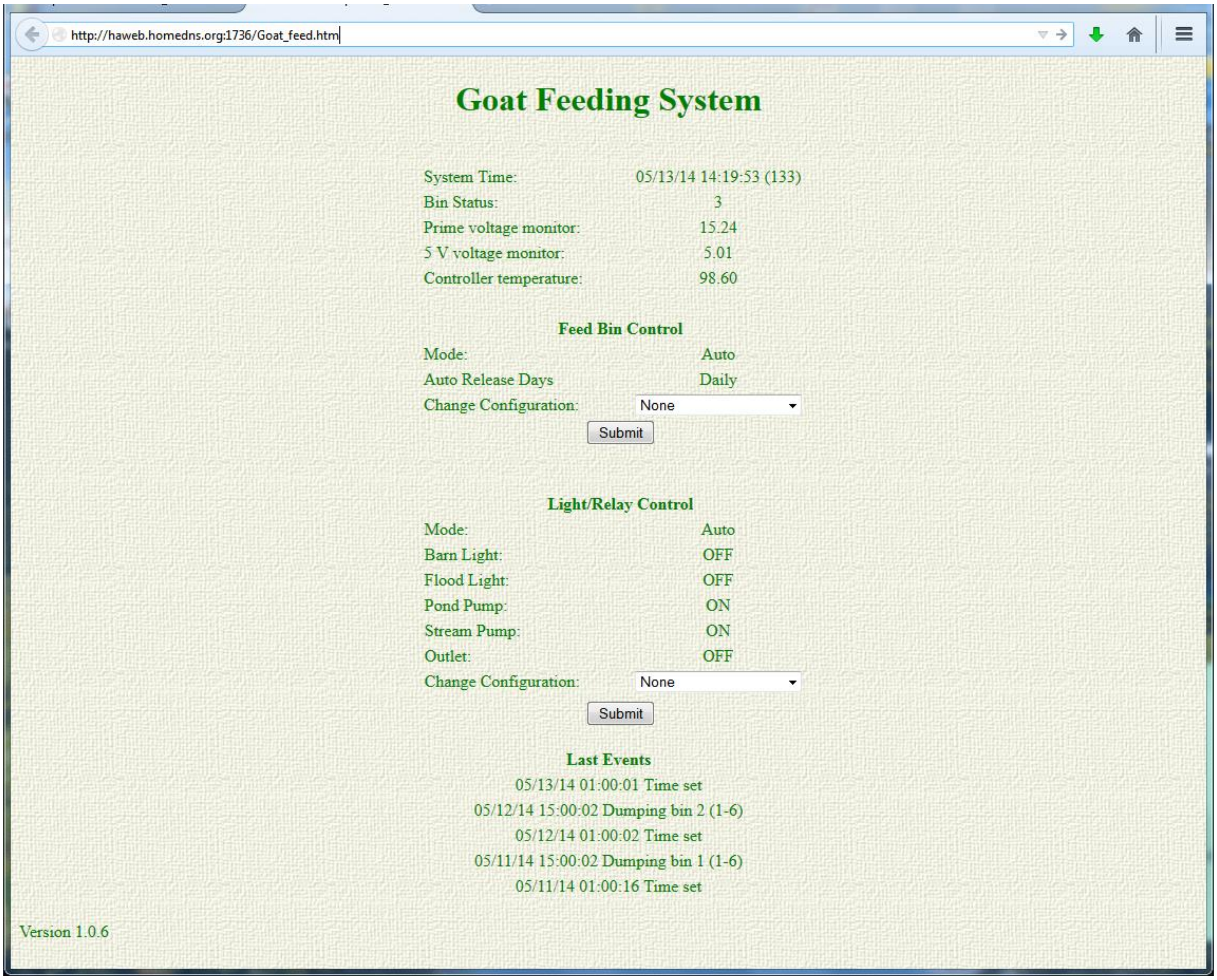

F211!
When making technical inquires. Please refer to this number & Cat. No.
JAMECO
102112

Figure 3 Photo of control electronics for the Animal Feeder System – The custom printed wiring board in the upper left hosts the chipKit Max32 controller and the WIZ550io module. The board in the middle contains the solid state relays that control the linear actuators. The terminal blocks on the bottom and on the right provide the interface for the AC wiring and bin sensor signals.



http://haweb.homedns.org:1736/Goat_feed.htm

Goat Feeding System

System Time:	05/13/14 14:19:53 (133)
Bin Status:	3
Prime voltage monitor:	15.24
5 V voltage monitor:	5.01
Controller temperature:	98.60

Feed Bin Control

Mode:	Auto
Auto Release Days	Daily
Change Configuration:	None

Light/Relay Control

Mode:	Auto
Barn Light:	OFF
Flood Light:	OFF
Pond Pump:	ON
Stream Pump:	ON
Outlet:	OFF
Change Configuration:	None

Last Events

```
05/13/14 01:00:01 Time set
05/12/14 15:00:02 Dumping bin 2 (1-6)
05/12/14 01:00:02 Time set
05/11/14 15:00:02 Dumping bin 1 (1-6)
05/11/14 01:00:16 Time set
```

Version 1.0.6

Figure 4 Control/Monitoring Web Page – This webpage is generated by the web server built into the control system. The web interface provides a quick way to monitor the operation of the system. It is also possible to manually control they system via this web page if desired.

Sample of Code

As mentioned previously, a significant portion of this project was to develop the interface driver between the Microchip TCPIP stack and the WIZ550io module which uses the W5500 Ethernet Controller. Fortunately, WIZnet does provide a driver for the W5200 device which was used as a starting point for this development, but there are significant differences between the SPI interface frame of the W5200 part and the W5500 part. In addition, the memory mapping between the two parts is significantly different. The W5200 maps the Common Registers, Socket Registers and the Rx/Tx buffer block into a linear address space. For example, the Socket 0 Registers are located between 0x4000 and 0x402e. The W5500 device uses a Block Select field in the SPI frame to select a memory, and the addressing within the memory is zero based. For example, the Socket 0 Registers are in block 0x01 between addresses 0x0000 and x0030. In an effort to utilize as much of the existing W5200 driver code as possible in the new W5500 driver, the internal operation was left for the most part untouched, and the differences were handled in the functions that implemented the SPI interface and the functions that interface with the device memory.

The code shown below is the new WriteReg function:

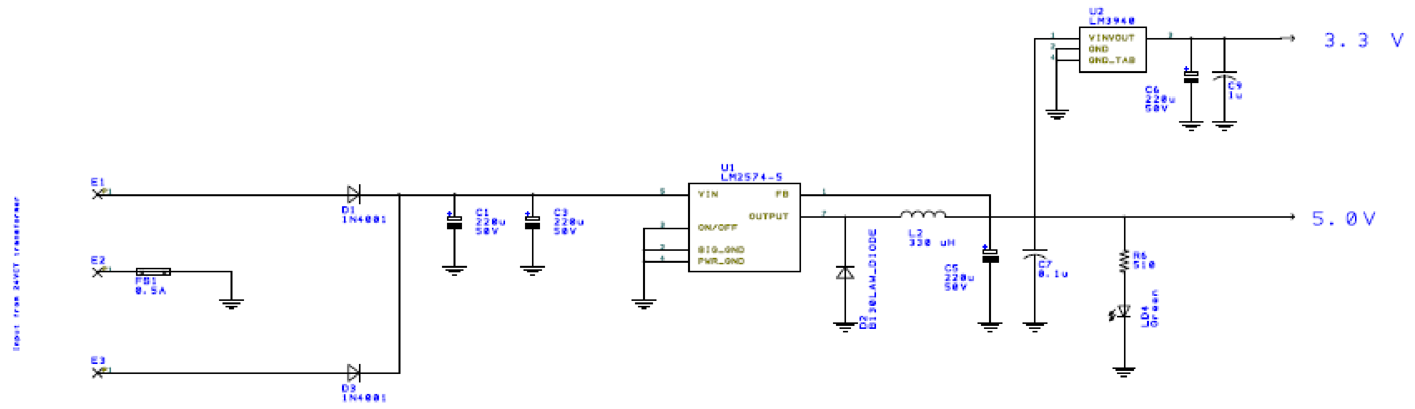
```
void WriteReg(BYTE BS, WORD Address, BYTE Data)
{
#if defined (__DISABLE_ALL_INT__)
    DINT();
#endif
    W5500_CS_IO = 0;

#if defined (__18CXX)  // For PIC18 Series

    W5500_SPI_SendByte((BYTE) ((Address & 0xFF00) >> 8));
    W5500_SPI_SendByte((BYTE) (Address & 0x00FF));
    W5500_SPI_SendByte(0x80);
    W5500_SPI_SendByte(0x01);
    W5500_SPI_SendByte(Data);

#else // for PIC32
    DWORD_VAL dwv;

    W5500_SPICON1bits.MODE32 = 1;
    if(BS == SOCKET_0_TX_BUFFER)
        dwv.w[1] = Address - TXSTART;
    else if(BS == SOCKET_0_RX_BUFFER)
        dwv.w[1] = Address - RXSTART;
    else
        dwv.w[1] = Address;

    dwv.w[0] = ((WORD) BS << 11) | 0x0400 | Data; //Second word is BS, R/W=1,OP=0, Data
    W5500_SPI_SendDword(dwv.Val); //Send 32 bits and read buffer
    W5500_SPICON1bits.MODE32 = 0;

#endif

    W5500_CS_IO = 1;
#if defined (__DISABLE_ALL_INT__)
    EINT();
#endif
}//end WriteReg
```

As mentioned earlier, the design also includes a watchdog function to prevent the linear actuators from being energized too long due to a software failure. The watchdog is implemented with a simple PIC16F688 processor. The watchdog monitors the drive signals for the linear actuators, and if they are active for more than 3 seconds it forces them to be de-energized to prevent them from getting too hot. The complete main() routine for this processor is shown below:

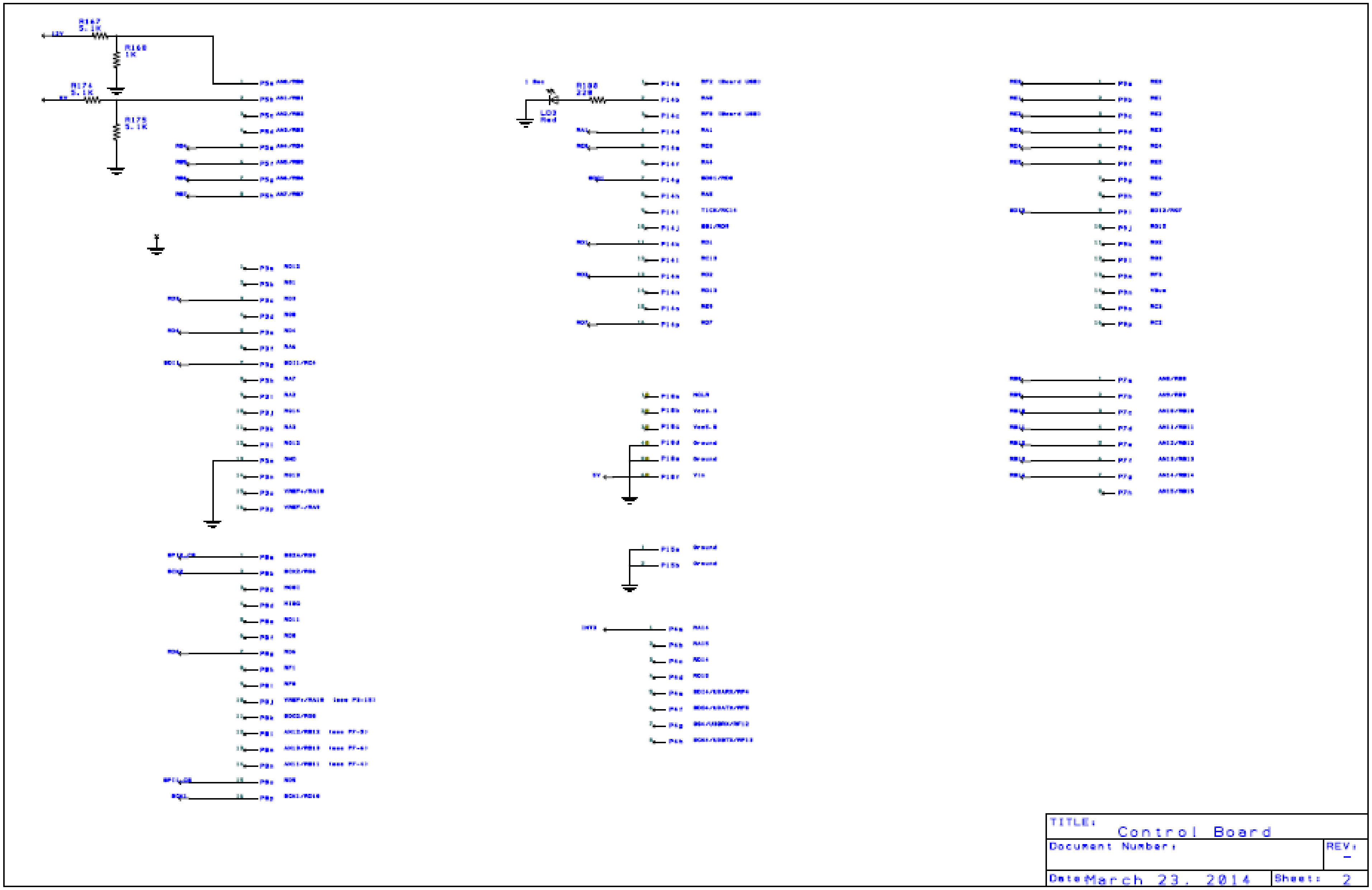
```
void main(void)
{
        Init();

        NewSecond = FALSE;
        TickCounter = 0;
        OnTime = 0;
        OffTime = 0;
        ShutDownFlag =  FALSE;

        //Set the Bin_enable true
        PORTC |= 0b00100000;

        while(1)
        {
                if(NewSecond == TRUE)
                {
                        //Do one second processing
                        NewSecond = FALSE;            //Clear the second flag
                        CLRWDT();                     //Kick the dog

                        //Determine if a solnoid is on
                        if(((PORTA & 0b00110100) | (PORTC & 0b00000111)) != 0)
                        {
                                //Solnoid is on so increment timer
                                OnTime++;
                                //Turn the LED on as long as an actuator is activated
                                PORTC |= 0b00001000;

                                if(OnTime >= ON_TIME_MAX)
                                {
                                        //Solnoid is ON too long so shut down
                                        ShutDownFlag =  TRUE;
                                        OnTime--;
                                }
                        }
                        else
                        {
                                OnTime = 0;
                                //Toggle the one second LED
                                PORTC ^= 0b00001000;
                        }

                        if(ShutDownFlag == TRUE)
                        {
                                //Set the Bin_enable false and turn the LED OFF
                                PORTC &= 0b11010111;
                                OffTime++;

                                if(OffTime >= OFF_TIME)
                                {
                                        //We have been off long enough so re-enable the solnoides
                                        ShutDownFlag = FALSE;
                                        OffTime = 0;
                                        OnTime = 0;

                                        //Set the Bin_enable true
                                        PORTC |= 0b00100000;
                                }
                        }

                }       //if(NewSecond)
        }       //while(1)
}
```

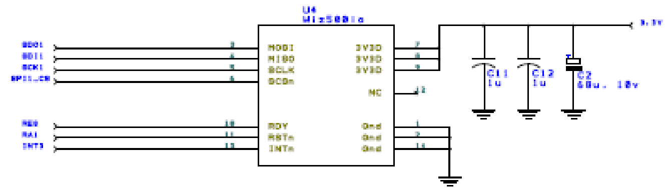
Control Board Schematic

Note: High resolution version of the control board schematic is available in “Control_board_schematic.pdf”



TITLE: Control Board		
Document Number:		REV: –
Date: March 24, 2014	Sheet: 1	


R167
5.1K
12V
R168
1K
R174
5.1K
R175
5.1K
R180
220
LD3
Red
TITLE:
Control Board
Document Number:
REV:
-
Date March 23, 2014
Sheet: 2



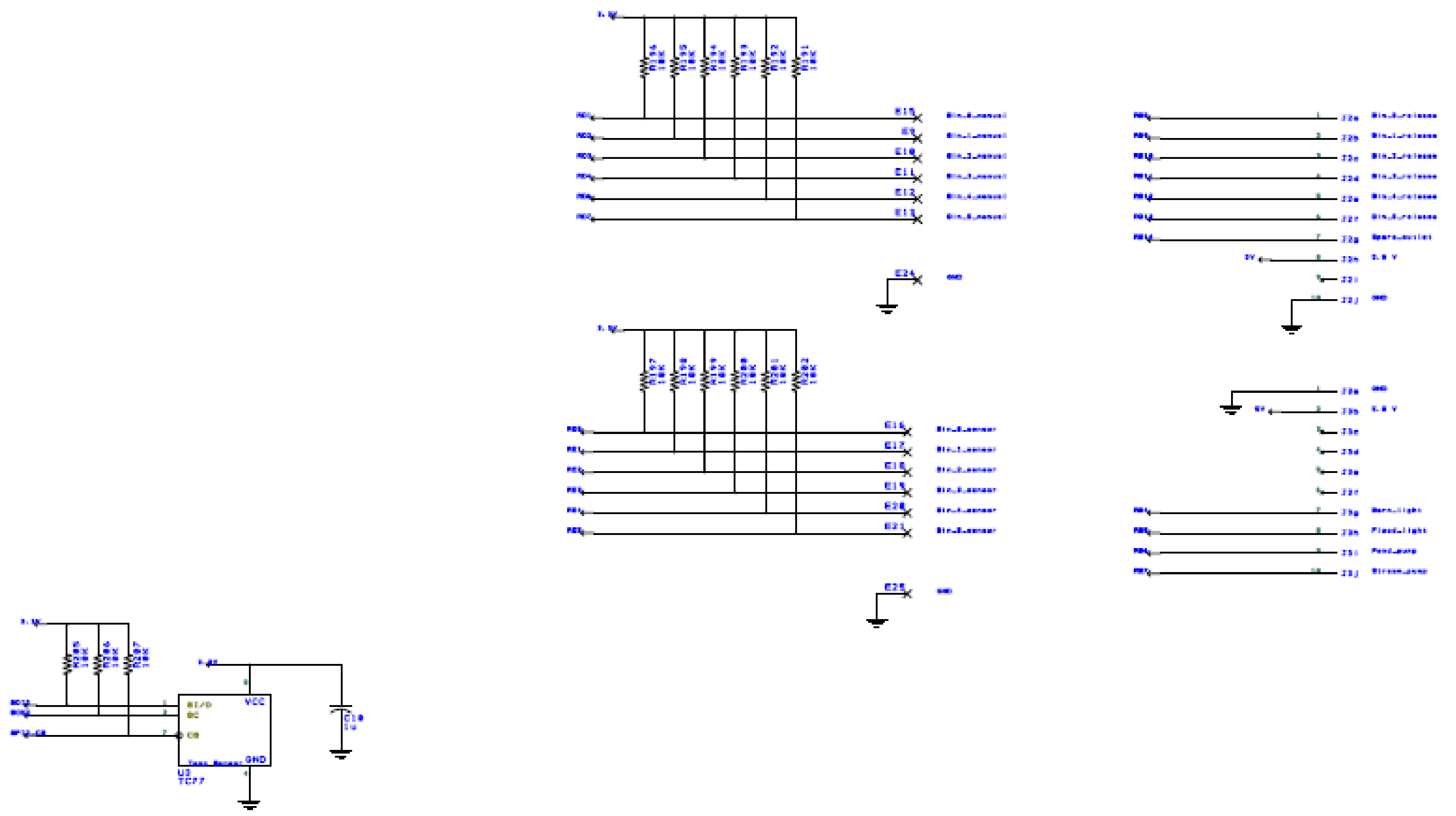
TITLE: Control Board

Document Number:

REV: -

Date: March 24, 2014

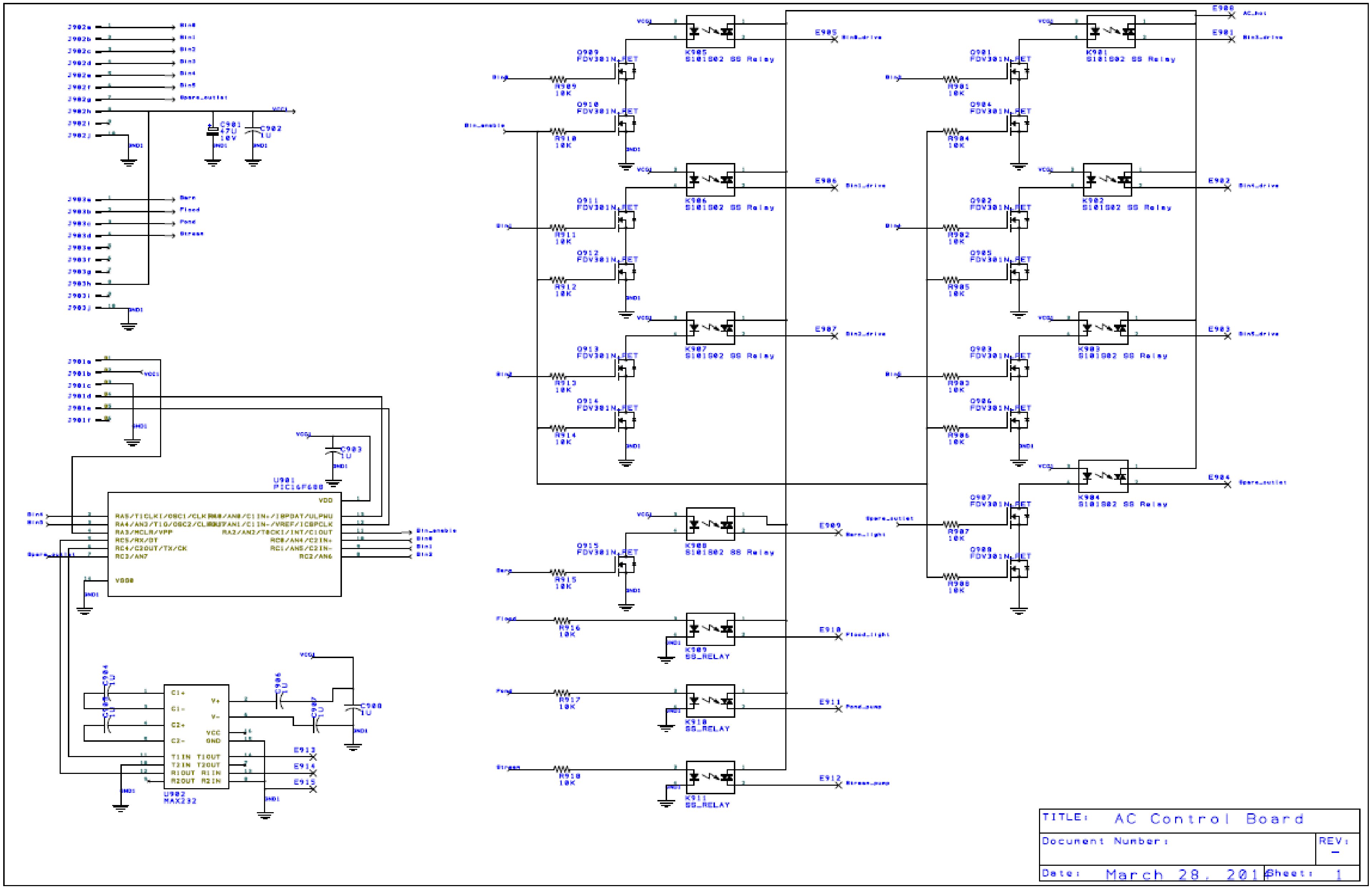
Sheet: 3



TITLE: Control Board		
Document Number:		REV: -
Date March 24, 2014	Sheet: 4	

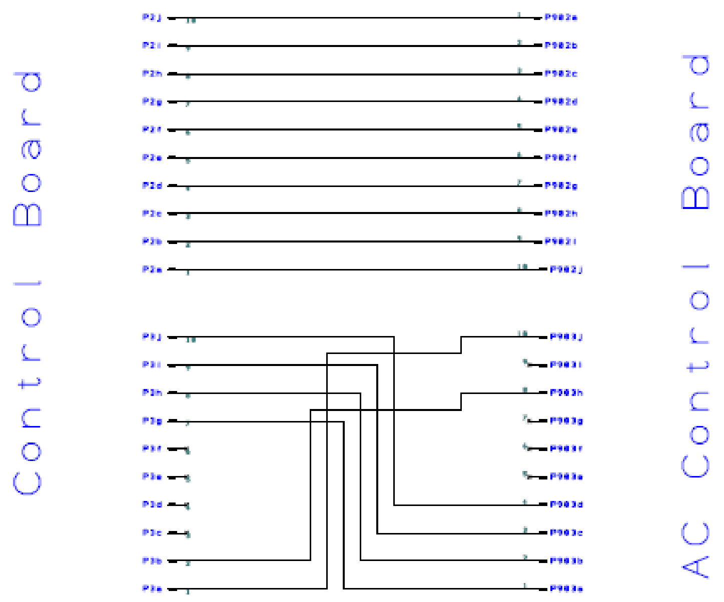
AC Control Board Schematic

Note: High resolution version of the AC Control Boar is available in "AC_control_board_schematic.pdf"

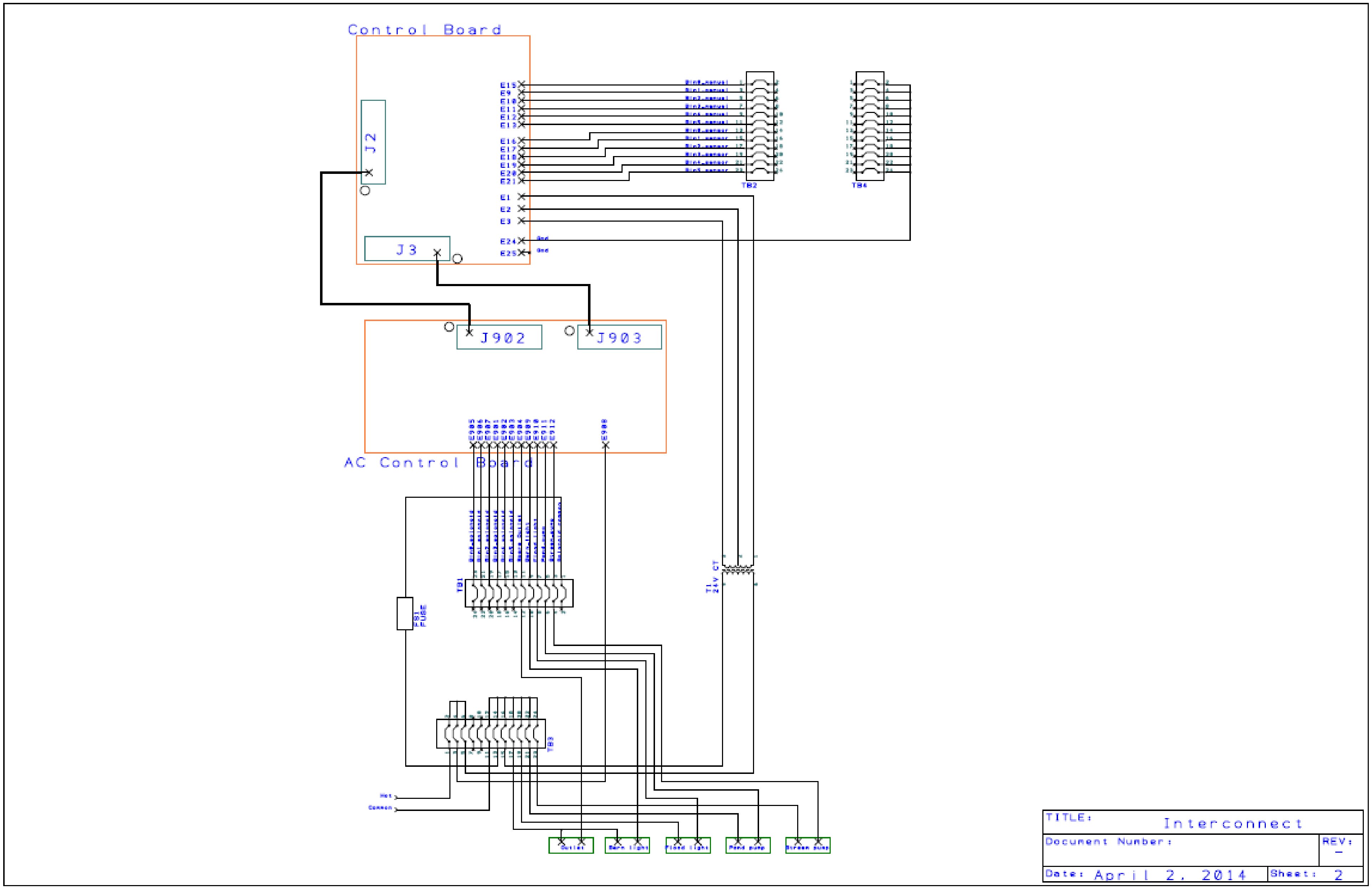

TITLE: AC Control Board
Document Number:
REV: -
Date: March 28, 201
Sheet: 1
U901 PIC16F688
U902 MAX232
K905 S101S02 SS Relay
K906 S101S02 SS Relay
K907 S101S02 SS Relay
K908 S101S02 SS Relay
K901 S101S02 SS Relay
K902 S101S02 SS Relay
K903 S101S02 SS Relay
K904 S101S02 SS Relay
K909 SS_RELAY
K910 SS_RELAY
K911 SS_RELAY
Q909 FDV301N_FET
Q910 FDV301N_FET
Q911 FDV301N_FET
Q912 FDV301N_FET
Q913 FDV301N_FET
Q914 FDV301N_FET
Q915 FDV301N_FET
Q901 FDV301N_FET
Q904 FDV301N_FET
Q902 FDV301N_FET
Q905 FDV301N_FET
Q903 FDV301N_FET
Q906 FDV301N_FET
Q907 FDV301N_FET
Q908 FDV301N_FET
C901 47U 10V
C902 1U
C903 1U
C908 1U
E900 AC_hot
E901
E902
E903
E904 Spare_outlet
E905
E906
E907
E909
E910 Flood_light
E911 Pond_pump
E912 Stream_pump
E913
E914
E915
VCC1
GND1
VDD
VSS
Bin_enable
Spare_outlet
Barn
Flood
Pond
Stream
R909 10K
R910 10K
R911 10K
R912 10K
R913 10K
R914 10K
R915 10K
R916 10K
R917 10K
R918 10K
R901 10K
R904 10K
R902 10K
R905 10K
R903 10K
R906 10K
R907 10K
R908 10K

Interconnect

Note: High resolution version of the interconnect is available in “Interconnect_diagram.pdf”



TITLE:	Interconnect		
Document Number:		REV: –	
Date:	April 2, 2014	Sheet:	1


Control Board
J2
J3
E15
E9
E10
E11
E12
E13
E16
E17
E18
E19
E20
E21
E1
E2
E3
E24
E25
Gnd
Gnd
Bin0_manual
Bin1_manual
Bin2_manual
Bin3_manual
Bin4_manual
Bin5_manual
Bin0_sensor
Bin1_sensor
Bin2_sensor
Bin3_sensor
Bin4_sensor
Bin5_sensor
TB2
TB4
J902
J903
AC Control Board
E900
TB1
FS1
FUSE
T1
24V CT
TB3
Hot
Common
Outlet
Barn light
Flood light
Pond pump
Stream pump
TITLE:
Interconnect
Document Number:
REV:
–
Date:
April 2, 2014
Sheet:
2