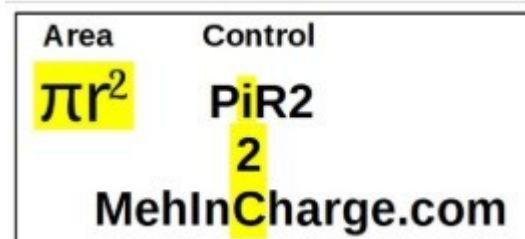


PiR2 CONTROLLER FUNCTIONALITY V8

as of 2020 D Apr 25

by David@ColeCanada.com

*** THIS ARTICLE (& HARDWARE & SOFTWARE) IS A WORK-IN-PROGRESS ***

**TABLE OF CONTENTS****Page**

| | |
|---|--|
| PURPOSE OF THE PiR2 CONTROLLER | |
| MIC WEBSITE FOR THE PiR2 | |
| PURPOSE OF THE PiR2 CONTROL PANEL | |
| REMOTE DESKTOP CONNECTION (FROM LOCAL WINDOWS LAPTOP) | |
| CONTROLLING THE PiR2 CONTROLLER FROM MehInCharge.com | |
| SUMMARY | |
| CONTROL PANEL FOR THE PiR2 CONTROLLER | |
| FUNCTIONALITY OF PiR2 CONTROL PANEL | |
| CONTROL PANEL ACTIONS | |
| SIMPLE or EMPTY CONTROL PANEL | |
| PiR2 CONTROLLER | |
| RASPBERRY Pi WITH THE FIRST PiR2A PROTOTYPE | |
| THE FIRST PiR2A PROTOTYPE (2020 D APR 25) | |
| CONNECTION TO THE GPIO PINS OF THE RASPBERRY Pi | |
| SOFTWARE COMPONENTS OF PiR2 | |
| VIRTUAL PiR2 CONTROLLER | |
| WORKING WITH THE PiR2 CONTROLLER | |
| INVOKING THE COMPONENTS OF THE PiR2 CONTROLLER | |
| INSTALLING THE PiR2 CONTROLLER SOFTWARE | |
| INVOKING THE PiR2 CONTROLLER SOFTWARE | |
| SYNCHRONIZING THE LOGS | |
| STARTING THE PiR2 CONTROL PANEL | |

APPENDICES (as of 2020 D Apr 25)

- A) PARAMETERS WITH GPIO #s AND PIN #s
- B) COMPLETE LIST OF ioDEVICES & PARAMETERS
- C) MAKING AN AUDIO WAV
- D) HANDLING OF EMAIL BY THE PiR2 CONTROLLER SOFTWARE
- E) OMNI-PURPOSE HIGH/LOW SENSOR CONDITIONER CIRCUIT
- F) EXTERNAL BOOKS AND SOURCES

PURPOSE OF THE PiR2 CONTROLLER

The PiR2 CONTROLLER (See Source 06) is designed to help a person monitor (and control) the environment in one area of a building, usually a house. The parameters monitored include measurements such as ambient temperature, ambient humidity, ambient light, audible sounds, nearby motion and many others (See Source 03 and the Appendix of this document). It is also possible to sense the ringing of a doorbell and to take a photo or video of an area in or around a house. Various on/off (1 bit) digital inputs can detect such actions as the opening of a window or door. A PiR2 CONTROLLER can also be used to control some devices in the house. For example, a gas fireplace can be lit or an alarm can be sounded through an optional attached amplifier and speaker or TV.

(The name PiR2 was chosen because it is the formula for the **area** of a circle. This is because the PiR2 is used to monitor and control the **area** around the PiR2. The PiR2 is an **area** controller.)

In this Internet age (2020), an email or text message can be automatically sent by the PiR2 CONTROLLER to a person's cell phone or laptop. In this manner, the owner of the house can be informed of any event. (Imagine receiving a text message informing you that the ambient temperature of your kitchen or the temperature of your freezer or kitchen refrigerator is too low.) The PiR2 CONTROLLER can even be used to charge a person's cell phone. A sleeping person can be warned (audibly) that his/her cell phone is not being charged. Using more than one PiR2 CONTROLLER will permit more than one area (i.e. multiple rooms) to be monitored. Each PiR2 CONTROLLER is connected to an inexpensive (\$35) Raspberry Pi computer. This computer can be attached to a cheap television set as a monitor. The photos, videos and audible alarms can be sent to this television set to inform people living in (or responsible for) the house of any unusual event.

Modern homes now have virtual voice assistants such as Google Assistant, Alexa, Cortana, Siri, etc (See Source 01). Using voice commands, they turn houses into smart homes where one can wirelessly turn on/off lights, fans etc. The PiR2 CONTROLLER can even "speak" a pre-recorded voice command through the Raspberry Pi's television speaker and thus control devices via these voice assistants. It would be interesting to know if the Google Assistant can be trained to communicate via the robotic voice generated by the existing espeak software! Read more later about the voice that espeak gives to the PiR2. This PiR2 CONTROLLER software is still free (as of 2020 D Apr 25).

MIC WEBSITE FOR THE PiR2

Each PiR2 CONTROLLER has its own PiR2 CONTROL PANEL. It would be inconvenient for the owner of a house to be forced to physically visit each PiR2 CONTROLLER in his house to view all the changes in its environment. Therefore, it made sense to design a central control panel that would amalgamate all the PiR2 control panels in the house. But it was just as simple to create a web site that would amalgamate all the house's control panels. Such a website would permit the owner to monitor and control the environment of his/her house from anywhere, while at work, travelling or on holidays. This website, called www.MehInCharge.com (See Source 05) has been created. Each PiR2 CONTROLLER can communicate with this website to store all of its measurements, be they sensed information or control information. Each PiR2 CONTROLLER is identified by the Serial Number of the Raspberry Pi computer to which it is attached. Each area being controlled by a PiR2 CONTROLLER is given a name (and location within the house). The information for each area is

linked to the area-name rather than to the Serial Number of the Pi. This allows for interchanging of hardware between areas without mixing up the exact location being monitored/controlled. Each house is identified by its post-office address. The post-office address will identify the latitude/longitude in most developed countries in the world.

PURPOSE OF THE PiR2 CONTROL PANEL

This document is written to primarily describe the future PiR2 CONTROL PANEL portion of the PiR2 CONTROLLER. At time of writing the first draft of this document, the PiR2 CONTROLLER is merely a tiny electronic circuit on a bread-board (more on this later). In April 2020, the first prototype of the PiR2A CONTROLLER was built. The basic software to operate the PiR2 CONTROLLER has been written and correctly transmits the sense and control information to the www.MehInCharge.com website. However the text in the PiR2 CONTROL PANEL does not yet drive the basic software. Therefore this document explains how the PiR2 CONTROL PANEL is designed to function, not how it actually functions at this time (as of 2020 D Apr 25).

The purpose of the PiR2 CONTROL PANEL is to permit the live operator to:

- 1) see some of the current parameters of the ioDevices as REPORTING values
- 2) see the actions (control actions or alarms) that will result from TRIGGERING parameter conditions
- 3) see the parameters that the controller is HOLDING to a set-point

Holding is done by adjusting the parameter using specific control actions to raise or lower it. (Of course, not all parameters can be controlled.) As with other controllers, the set point is controlled within a range (as shown under the words "Set Point"). This range is defined as a % which is equal to the rightmost 2 digits of the number shown above the "Set Point". This is usually 4%.

In each case, the current (most recent) value of each sensed parameter is displayed. The current value of each controlled parameter is also displayed. The control panel is refreshed at the shown REFRESH RATE (on text row 5).

Various PiR2 CONTROL COMMANDS can be issued by a live person i.e. the operator, by typing the command into the text box labelled "PiR2 Controller Command" and clicking on the "SEND" button. Examples of such commands are:

```
acquire:procTemp
digOut0:1
D_imageCamera:D_backYard?.gif
audioTV0:alarmA.wav
redLED:1
```

A PiR2 control command names the parameter-name followed by ":" which is followed by the value to send as the control. A PiR2 sense command begins with the word "acquire" then ":" then the parameter-name. Note that every parameter-name is unique. Parameter-names do not include a "_". This permits the specification of a parameter-name of an ioDevice on another PiR2 Controller. Each PiR2 CONTROLLER is assigned a unique areaCode which is a letter designating the piArea that it is controlling. When any piArea word is assigned to an area of a house, the areaCode must also be

assigned to the area. When a parameter is referenced, another area can be indicated. This is done by preceding the parameter by the areaCode followed by a “_”. In this manner it will be possible for the control Panel of one PiR2 CONTROLLER to control an ioDevice for another area by preceding the distant parameter-name with its distant piArea name's unique prefix followed by “_”. For example A_imageCamera will refer to the imageCamera device for piArea with areaCode "A". If no areaCode is specified, the parameter-name for the current PiR2 CONTROLLER will be presumed.

REMOTE DESKTOP CONNECTION (FROM LOCAL WINDOWS LAPTOP)

Windows-10 provides a facility called Remote Desktop Connection (RDP). It is a program found under Windows Accessories. It can be used to control the "desktop" of the Raspberry Pi computer from the "desktop" of a Windows-10 laptop or PC. To do this, the Raspberry Pi and the Windows-10 machine must be connected to the same in-house router. This can be used to control a “headless” Raspberry Pi. A “headless” computer is one that is not equipped with the following accessories: mouse, keyboard, monitor. The RDP facility can be used to "remotely" control a minimally configured Raspberry Pi and PiR2 CONTROLLER. But to do this, one must know the IP address and a user name and password for the Raspberry Pi. Unknowingly, a user might try to start a second instance of the PiR2 software on a Raspberry Pi. This is not allowed and the PiR2 software will not allow this.

CONTROLLING THE PiR2 CONTROLLER FROM MehInCharge.com

Some very limited control of the PiR2 CONTROLLER is possible by logging into the www.MehInCharge.com (MIC) website from any laptop, pc, iPad, tablet or cellphone. This is done by using the menu in the top-right corner of the MIC website. To do this, first, click on Control button under “_ACTION_”. Then type any PiR2 command into the text box labelled "command". Then click on the Upload button. As of 2020 D Apr 25, only a small number of commands will work correctly. One such function is "piArea:Kitchen". This will assign "Kitchen" to the piArea. But the execution of each command typed into MIC is deferred. Each command is put in a "command queue" that will only be transmitted to the PiR2 controller the next time that the PiR2 CONTROLLER is used to visit the www.MehInCharge.com website and to synchronize its data with the website. After completing the synchronization of the log and command data, the commands in the "command queue" will be automatically processed by the PiR2 CONTROLLER.

Some examples of commands that can be performed in this manner (as of 2020 D Apr 25) are:

piArea:Kitchen

This states that the PiR2 is controlling the kitchen area

audioTV0:Hey_Google_Turn_On_The_Fan.wav

This tells the Google Assistant to turn on the fan linked to it.

audioTV0:Coupe.mp3

This plays the song “Little Deuce Coupe”

acquire:tempId00

This reads and records the temperature from thermometer number “00”

acquire:piSN

This reads and records the Serial Number of the Raspberry Pi computer

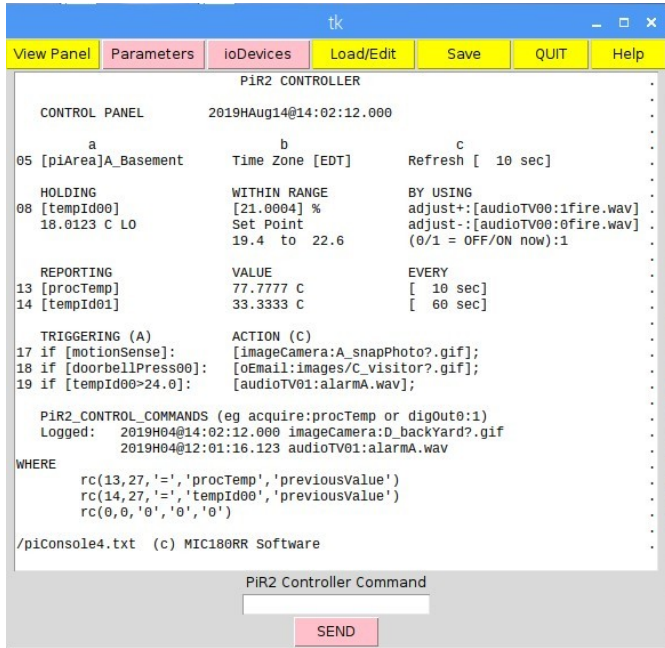
acquire:piIP

This reads and records the IP address being used by the Raspberry Pi computer

SUMMARY

The PiR2 CONTROLLER is a hardware controller implemented using software defined by MIC180RR Corp. (belonging to David@ColeCanada.com). It is seen as an on-screen CONTROL PANEL that can be used by a live operator (a person) of the Raspberry Pi computer. The live operator has bi-directional communication with the CONTROL PANEL. However the PiR2 CONTROLLER is designed to operate automatically unattended most of the time (up to 100% of the time) without any live operator being present. This CONTROL PANEL communication will soon be invoked by a Python (see Source 07) script containing buttons and text (similar to those of HTML). Each button can be pressed to dynamically perform a software action. Some of the text information displayed on the control panel is continually dynamically updated by the PiR2 controller. Other text can only be dynamically changed by the live operator. Other text can only be changed by specifying that a new version of the console control panel text file be loaded by the python routine.

CONTROL PANEL for the PiR2 CONTROLLER



The control panel is comprised of the following components:

a) a menu across the top containing the following buttons which can be clicked by the live operator:

Live Panel

"Live Panel" is clicked to begin the automatic operation of the PiR2 Control Panel as displayed.

ioDevices

"ioDevices" lists the input/output devices that are operated by the PiR2 Controller

Parameters

"Parameters" lists the one (or more) input/output parameters that each physical ioDevice (in the attached PiR2 controller) can read (i.e. sense) or write (i.e. control).

LoadEdit

"LoadEdit" permits the live operator to select the "initial" text file that will be loaded into the PiR2 control panel. Once loaded, it can be immediately be edited by the live operator. If it is edited, the new version should be Saved by the live operator

Save

"Save" permits the live operator to save the current text file with a new name.

QUIT

"QUIT" stops running the Python CONTROL PANEL routine

HELP

"HELP" describes the operation, control and functionality of the PiR2 CONTROL PANEL

b) a text file named piConsole4.txt where 4 is the version. This text file is automatically updated to display the most recent values of the parameters sensed (measured) by the PiR2 CONTROLLER. It also automatically invokes control parameters. The mechanism used to provide this functionality is described later in this document. Note that the first row of the text file is assigned row # 0 and the first column of each row is assigned column # 0.

c) a button at the bottom labelled "SEND" which can be clicked by the live operator after a PiR2 CONTROLLER command has been typed into the text box above the SEND button.

FUNCTIONALITY OF PiR2 CONTROL PANEL

When the "Live Panel" (formerly named "View Panel") button is activated, the Python routine then does the following:

1) analyses the text that appears in the CONTROL PANEL, noting the number of each of the following types of actions that must be performed:

HOLDING

REPORTING

TRIGGERING

LOGGING recent commands

NOTING the WHERE clauses

2) Noting exactly which parameters must be used in conjunction with the above actions.

3) Noting the intervals when the above actions are to be performed

4) Creating code sequences to implement the controls required for each of the actions

5) Identifying errors (e.g. non-existent files or invalid parameters that are referenced)

6) Immediately notifying the live operator of any errors or
if no errors exist,

7) Displaying the piSN (Serial Number of the Raspberry Pi)

8) Activating only the Load/Edit, QUIT and SEND buttons,

9) Beginning execution of the code sequences.

An error is triggered if the last WHERE clause is not "rc(0,0,'0','0','0')". After clicking on "Live Panel", if the live operator clicks on Load/Edit, he is permitted to modify any field that is enclosed within square braces. During the early operation of this CONTROL PANEL, very few fields will be enclosed within square braces.

Note that every readable parameter will be refreshed, recorded and uploaded to MehInCharge.com after each interval labelled "Refresh". Every time a control action occurs, it will be recorded and uploaded to MehInCharge.com . The system will also record every command issued by a live operator using the CONTROL PANEL or by a person "legally" logged into this PiR2 CONTROLLER at MehInCharge.com . This CONTROL PANEL does not verify the results of any downloads or changes made while in "Live Panel".

CONTROL PANEL ACTIONS

HOLDING

This type of control is to perform the standard closed loop control normally used in any process control system. The HOLDING process is describe in more detail below. Errors may occur, so it is wise to include a trigger mechanism that will notify a responsible person of any alarming situation. The HOLDING of one or more parameters is optional (not necessary).

REPORTING

This type of control is a simple frequent display of any parameter values that are of special significance. Note that all parameters will be read at the "Refresh" rate, so it is not necessary to report any parameters more frequently.

TRIGGERING

This type of control can be used to produce Alarms. All triggered alarms are logged to MehInCharge.com , even the least significant alarms. It is not necessary to trigger any alarms.

LOGGING Control Actions

This type of control is not optional and is continually ongoing. There are 2 sources of control actions, automatic (by the PiR2 CONTROLLER) and those requested by a human operator. The operator can be either a live operator controlling the Raspberry Pi directly, an RDC (Remote Desktop Connection) operator (on a device connected to the same router as the Pi) or a remote operator logged into MehInCharge.com from anywhere. Regardless of the location of the live operator, all control actions are logged. Each sensing of a parameter is also logged. The readings of all parameters that are sensed are logged and are sent to MehInCharge.com. Parameters that are not mentioned on the CONTROL PANEL will not appear on the CONTROL PANEL, but they are always logged.

NOTING WHERE

Each parameter that appears in the NOTING WHERE portion of the PiR2 CONTROL PANEL will be repetitively updated on the PiR2 CONTROL PANEL. This is done so that the human operator can see the most recent value of the parameters to be displayed. The PiR2 CONTROL PANEL updates the text at the row/column designated for the specified parameter.

SIMPLE or EMPTY CONTROL PANEL

An empty CONTROL PANEL text file need only contain a small amount of the text shown above. The only essential text is in row 5, row 21 and row 27 of the text shown previously). The row numbers are only significant for the WHERE clauses. The minimum (i.e. empty) CONTROL PANEL would appear as follows

```

-----
                PiR2 CONTROLLER

CONTROL PANEL      dateTime:2019HAug14@14:02:12.000
                  piSN:00000000000000000000
                a                b                c
05 piArea:A_Basement    timeZone:EDT    refreshRate:[ 10 ] sec

07 Recent PiR2_CONTROL_COMMANDS (eg acquire:procTemp or digOut0:1)

09 WHERE
10   rc(0,0,'0','0','0')

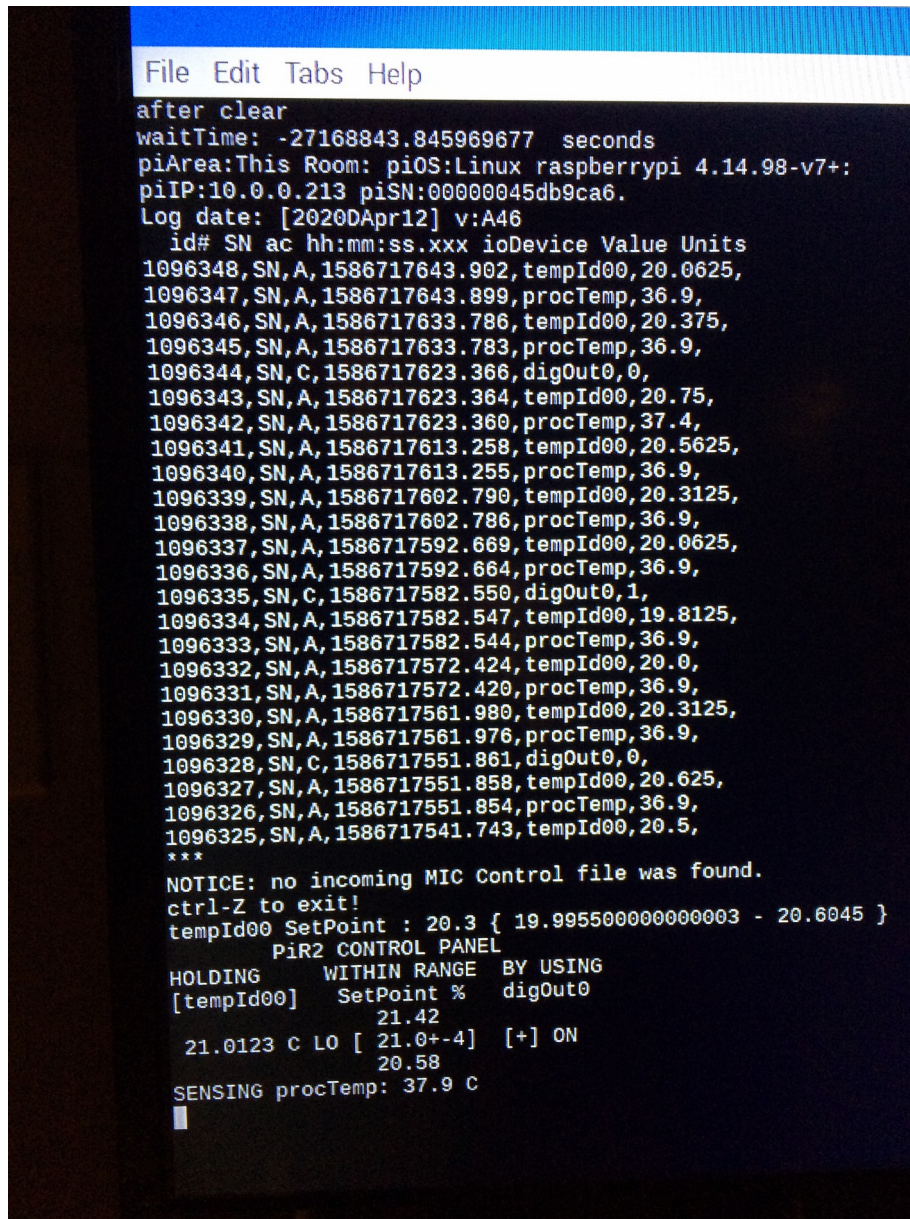
/piConsole1.txt (c) MIC180RR Software
-----

```

A PiR2 CONTROL PANEL using this text file will read every parameter every 10 seconds, and will log all of the readings. If the refreshRate is set to be too frequent, the Raspberry Pi will become overloaded and will either fail or the available storage will be exceeded. A good value to begin with is 60 sec. Every time the PiR2 CONTROLLER is synchronized with MehInCharge.com , portions of the log will be uploaded to MehInCharge.com . Note that any commands that are in the MIC command queue will be executed after synchronization. Such commands will also be included in the log sent to MehInCharge.com .

PIR2 CONTROLLER

The PiR2 CONTROLLER includes the PiR2A hardware and the PiR2 Python software. It is run by starting up the Raspberry Pi computer. Once started the Pi should display the Raspbian Desktop. To start up the PiR2 CONTROLLER, start the Pi Terminal and type “sh pir2.sh”. To skip verbose info, hit Enter. After about 20 seconds, hit Enter again. Then the operator can optionally type in any PiR2 command. Each command will be immediately executed. After the last command, hit Enter. Then the PiR2 CONTROLLER should start running and will show a screen similar to the following:



```

File Edit Tabs Help
after clear
waitTime: -27168843.845969677 seconds
piArea:This Room: piOS:Linux raspberrypi 4.14.98-v7+
piIP:10.0.0.213 piSN:00000045db9ca6.
Log date: [2020Apr12] v:A46
  id# SN ac hh:mm:ss.xxx ioDevice Value Units
1096348,SN,A,1586717643.902,tempId00,20.0625,
1096347,SN,A,1586717643.899,procTemp,36.9,
1096346,SN,A,1586717633.786,tempId00,20.375,
1096345,SN,A,1586717633.783,procTemp,36.9,
1096344,SN,C,1586717623.366,digOut0,0,
1096343,SN,A,1586717623.364,tempId00,20.75,
1096342,SN,A,1586717623.360,procTemp,37.4,
1096341,SN,A,1586717613.258,tempId00,20.5625,
1096340,SN,A,1586717613.255,procTemp,36.9,
1096339,SN,A,1586717602.790,tempId00,20.3125,
1096338,SN,A,1586717602.786,procTemp,36.9,
1096337,SN,A,1586717592.669,tempId00,20.0625,
1096336,SN,A,1586717592.664,procTemp,36.9,
1096335,SN,C,1586717582.550,digOut0,1,
1096334,SN,A,1586717582.547,tempId00,19.8125,
1096333,SN,A,1586717582.544,procTemp,36.9,
1096332,SN,A,1586717572.424,tempId00,20.0,
1096331,SN,A,1586717572.420,procTemp,36.9,
1096330,SN,A,1586717561.980,tempId00,20.3125,
1096329,SN,A,1586717561.976,procTemp,36.9,
1096328,SN,C,1586717551.861,digOut0,0,
1096327,SN,A,1586717551.858,tempId00,20.625,
1096326,SN,A,1586717551.854,procTemp,36.9,
1096325,SN,A,1586717541.743,tempId00,20.5,
***
NOTICE: no incoming MIC Control file was found.
ctrl-Z to exit!
tempId00 SetPoint : 20.3 { 19.995500000000003 - 20.6045 }
  PiR2 CONTROL PANEL
HOLDING      WITHIN RANGE  BY USING
[tempId00]  SetPoint %   digOut0
           21.42
21.0123 C LO [ 21.0+-4]  [+] ON
           20.58
SENSING procTemp: 37.9 C

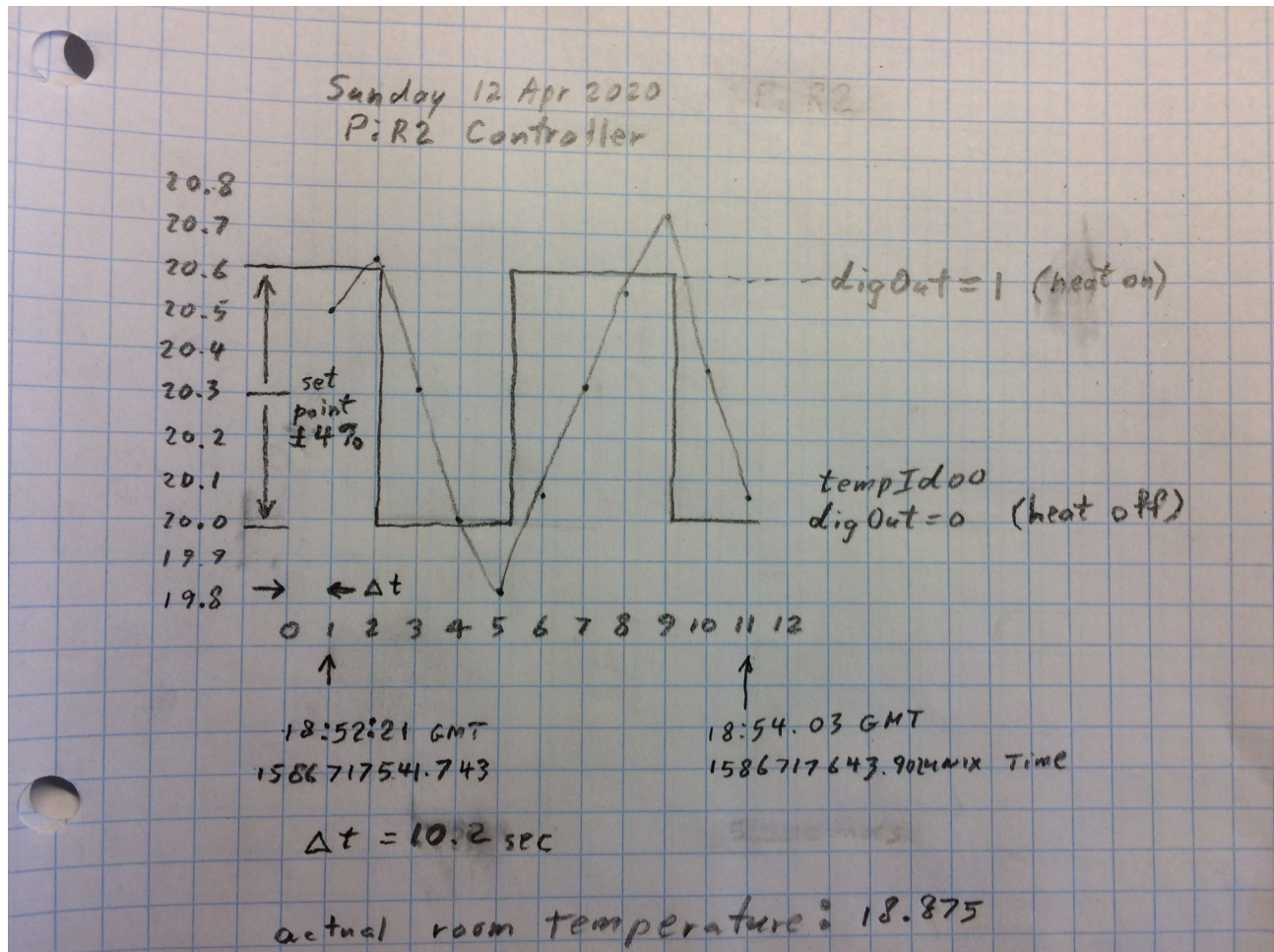
```

The above image displays the PiR2 screen which is automatically refreshed to include the past 100 seconds of activity. Note that the above log has recorded many values of the 3 PiR2 parameters: tempId00, procTemp and occasionally digOut0. At the bottom it says that it is controlling the tempId00 within 3% of its set point of 20.3 degrees Centigrade. The PiR2 CONTROL PANEL at the very bottom

does not contain up-to-date information and is merely an initial attempt to explain what the PiR2 CONTROLLER is meant to be doing. Eventually it will display live data.

To demonstrate how the PiR2 can control the temperature in a room, a small resistor is mounted close to (just above) the tiny (3mm x 3mm) IC chip of the TMP102 (explained in more detail below). This resistor heats up when the digOut0 (control) is set to a value of 1 by the PiR2 CONTROLLER and its software. The graph below shows the closed loop control cycle accomplished by the PiR2 CONTROLLER.

The following graph (of the above log) explains the control that is accomplished:

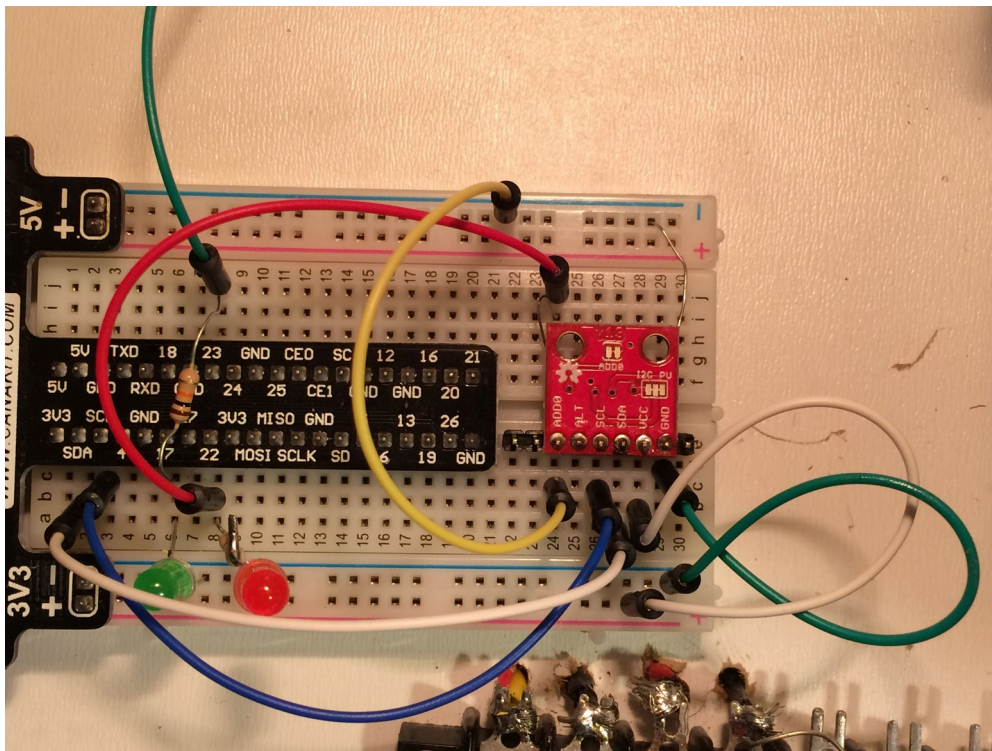


Only 2 parameters appear in the graph. The horizontal axis is the number of PiR2 log cycles (from 0 to 12). Each log cycle is 10.2 seconds. The equivalent Unix Time and GMT (Greenwich Mean Time) are shown. The date is shown at the top of the graph. The vertical axis is the tempId00 (from the TMP102 sensor) that the PiR2 CONTROLLER is using to measure the “ambient Temperature” in degrees Centigrade. The digOut0 (heat) is also shown on the vertical axis.

During cycle 1 the (ambient) temperature is 20.5 C. The Set Point of the tempId00 is 20.3 C +/- 3% (i.e. a range between 19.99 C and 20.60 C). During cycle 2, the temperature is 20.625 which has risen.

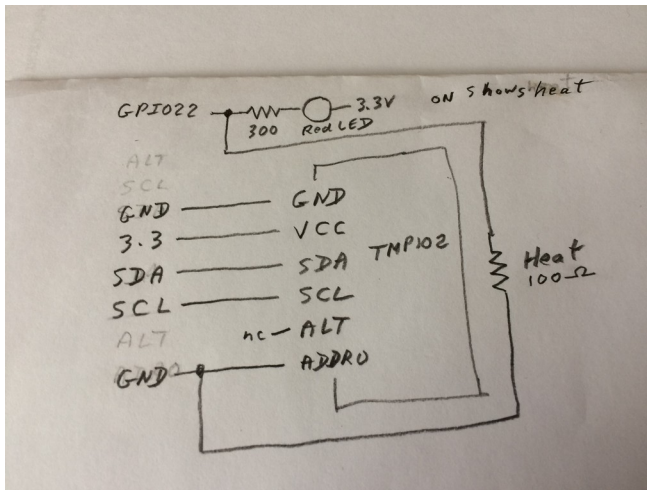
This indicates that the heat is probably currently ON. This temperature is above the range, so digOut0 is set to 0 by the PiR2 CONTROLLER, which turns off the resistor which stops heating up the TMP102 sensor. During cycle 3, the temperature has dropped to 20.3125 C. The temperature continues to drop during cycles 4 and 5. In cycle 5, the temperature has dropped to 19.8125 which is below the lower limit of the range, so digOut0 is set to 1 which begins to heat up the resistor again. During cycle 6 the temperature reaches 20.0625 which is in range again. But digOut0 is still 1, so the resistor continues to heat up. The temperature continues to rise during cycles 7, 8 and 9. During cycle 9, the temperature reaches 20.75 C which is above the range. So the PiR2 CONTROLLER sets digOut0 to 0 (off) which turns off the (heat) resistor allowing the temperature of the TMP102 to start dropping again. This completes the full temperature control cycle.

A photo of the PiR2 CONTROLLER breadboard prototype appears below:



The TMP102 temperature sensor of PiR2 CONTROLLER breadboard prototype is shown above. It is the tiny (1cm x 1cm) red circuit board on the right in the photo above. The actual TMP102 IC chip (tempId00) is not visible because it is mounted on the under-side of the small red circuit board on the right. The “heater”, a 100 ohm resistor, is mounted under the TMP102 chip so it is not very visible. It is hidden by the TMP102 but one of its leads can be seen connected to ground on the breadboard. The other lead of the resistor is connected to slot 24 of the breadboard. Slot 24 is connected to GPIO22 of the “Wedge” which is connected to the GPIO pins (shown on the next page) of the Raspberry Pi computer. GPIO22 is digOut0. Different GPIO pin numbers are used on the final prototype of the PiR2 CONTROLLER.

An early schematic of the TMP102 part of the PiR2 CONTROLLER prototype appears below:



It is interesting to note that most of the pin-outs of the TMP102 match the Raspberry Pi.

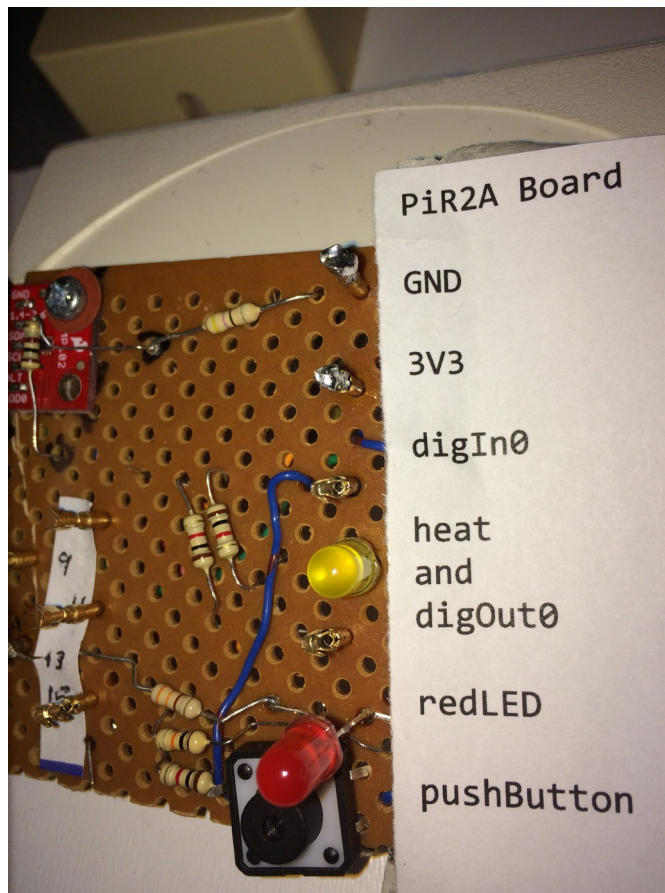
RASPBERRY Pi WITH THE FIRST PiR2A PROTOTYPE

The first prototype of the PiR2A (not on a breadboard) is shown below on the right. A ribbon cable connects it to the GPIO pins of the Raspberry Pi 3 which is on the left.



The first PiR2A controller (shown on the right above) is equipped with a single row of male header pins. These header pins connect into one side of the grey ribbon cable that is connected to the Raspberry Pi (on the left). Some physical pins (those with even numbers 2 to 40) are not used. Only pins 1, 3, 5, 7, 9, 11, 13, 15, and 25 are used. They were chosen so that only a single set of header pins are needed on the PiR2A. This choice of pins makes the PiR2A compatible with the earliest Raspberry Pi Model 2.

A larger photo of the PiR2A controller prototype is shown on the next page.

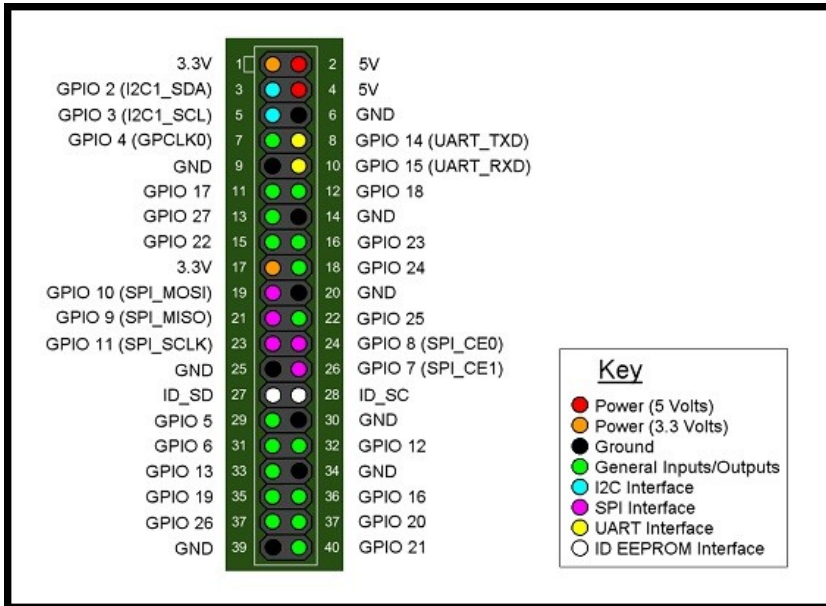
THE FIRST PiR2A PROTOTYPE (2020 D APR 25)

The header pins that connect this PiR2 CONTROLLER to the Raspberry Pi cannot be seen. They are located immediately to the left of the photo. The tiny red TMP102 board is located in the top left corner. A 100 ohm resistor sits directly on top of the tiny IC chip and is used as a dummy heater that can heat up the TMP102 IC thermometer. This is done to demonstrate how the PiR2 can be used to control the temperature in the area of the Pi. The redLED can permit the PiR2 to announce a simple visual alarm. Other announcement methods are also possible. The push Button in the bottom right corner permits the operator to provide either an answer (Yes is Down / No is Up) to a question or to get the attention of the PiR2A CONTROLLER. The yellow LED lights up when the digOut0 signal goes high. This does not signal an alarm, the yellow LED is lit when the digOut0 is turned on (and when the heat resistor is optionally turned on) . The heat resistor heats up when the digOut0 signal goes high unless it is optionally disabled. If the heat resistor is disabled from the circuit, it will permit the TMP102 to sense the true temperature of the area. The heat resistor is disabled by cutting a trace on the PiR2A board. The digIn0 can be set to 0v or 3.3v by an external source. The digIn0 be read at any time by the PiR2A controller. Alternatively the pushButton can be pushed to apply 3.3v to digIn0.

This prototype PiR2A board demonstrates how the Raspberry Pi can be used to sense digital and analogue signals. It can also output digital signals electronically or visually via the LED. The Raspberry Pi can also output alarms via the television monitor and/or the audio output jack on the Raspberry Pi.

CONNECTION TO THE GPIO PINS OF THE RASPBERRY Pi

The TMP102 is connected to the Raspberry Pi via I2C (see Source 04) using SDA (GPIO2) and SCL (GPIO3). A diagram of the GPIO pins (See Sources 02 and 03) of the Raspberry Pi 3 is shown immediately below. Those of the earlier Pi 2 are shown further below. Software written for Pi models 2, 3 and 4 is compatible with later hardware modules, but is NOT backward compatible.

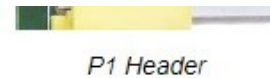


GPIO pinouts for Rev 3 Pi



GPIO pinouts for Rev 2 Pi

- red ones are +ve power (3V3 or 5V)
- black ones are -ve ground
- yellow ones are all dedicated general purpose I/O ports (OK, 18 does PWM as well, but forget that for now).



The rest can all be used as GPIO ports, but do have other functions too. If you need 8 or less ports, it's best to use the yellow ones because you're less likely to have a conflict with other things. A quick rundown of what the others are...

- greeny/grey – i²c interface
- light grey – UART (serial port)
- orange – SPI (Serial Peripheral Interface)

SOFTWARE COMPONENTS OF PiR2

The complete PiR2 CONTROLLER software has the following 3 components:

- a) PiR2A (PiR2main.py current version: 48 as of 2020 D Apr 25)
- b) www.MehInCharge.com
- c) PiR2ControlPanel (panelGUI.py)

Each component is run independently as a separate thread or instance. The main thread named PiR2main.py must be running in order for the PiR2 CONTROLLER to function. The website MehInCharge.com can be accessed by anyone from any personal device that is on the web. But at this time, the live operator of the PiR2 CONTROLLER must log into the MehInCharge website from his/her Raspberry Pi in order to synchronize the current PiR2 logs on the Raspberry Pi with the full log stored at MehInCharge.com. A lesser part of the synchronization occurs in the opposite direction. This is the transfer of latent commands from the command queue at the MehInCharge website to the PiR2 Controller. At this time (2020 D Apr 25) the live operator of the each PiR2 CONTROLLER must initiate the synchronization between the PiR2 CONTROLLER and the MehInCharge website. Eventually the synchronization will occur automatically. The PiR2 CONTROL PANEL is the future face of the PiR2 CONTROLLER. If the synchronization is never initiated, all of the logs will remain in the Raspberry Pi to which the PiR2 CONTROLLER is connected. The full log is actually a collection of PiR2 logs that are amalgamated at MehInCharge.com. A separate log is created in the Raspberry Pi each time the PiR2 CONTROLLER software is started up. But no mechanism exists to view or manipulate these logs using PiR2 software in the Raspberry Pi. To do this, powerful software has been created at the MehInCharge website. These logs are hard to understand unless they are sent to the MehInCharge website. Today (2020 D Apr 25), the PiR2 CONTROLLER fully functions even though the PiR2 CONTROL PANEL is not running. It also functions fully, even if it is never synchronized with the MehInCharge website. Eventually, the PiR2 CONTROL PANEL will be the primary interface between the PiR2 CONTROLLER and its live operator. Then, the PiR2main.py will be automatically started and will be running as the invisible heart of the PiR2 CONTROLLER. When this happens, the live operator will only need to use the PiR2 CONTROL PANEL and the MehInCharge.com website.

VIRTUAL PiR2 CONTROLLER

The PiR2 software has been designed to fully operate whether or not an actual PiR2 CONTROLLER is present. This is possible because one changing parameter of the Pi can be sensed. The procTemp (temperature of the Raspberry Pi processor) can be accessed whether or not a PiR2 CONTROLLER is present. This situation is said to be using a virtual PiR2 CONTROLLER. The PiR2 software can detect whether or not a physical PiR2 CONTROLLER e.g. the PiR2A is present. In the case of a virtual PiR2 CONTROLLER, the absence of other sensors and actuators (that are mainly in the PiR2 Controller) drastically limit the functionality of the PiR2 CONTROLLER. Other non A/C ioDevices are available to the PiR2 CONTROLLER software. Examples are the audioTV speaker, the imageTV screen and the audioJack into which earphones can be plugged. But interestingly enough, the Raspberry Pi can be run in a “headless” manner (with no monitor nor keyboard nor mouse). One can communicate with a headless Raspberry Pi using any computer attached to the same router as the Raspberry Pi (as described earlier). This permits a person to purchase a Raspberry Pi computer and power supply, plug it in with no TV, no mouse and no keyboard. Then connect it to the web via an

Ethernet cable, use a local computer to download and install the PiR2 software and begin to monitor and log the temperature of the processor in the Raspberry Pi. The main difficulty when doing this is apparently to determine the IP address that is assigned to the Pi by the router. The data logged in this manner can even be uploaded to the www.MehInCharge.com website. It is, of course, a lot easier to do this if a TV, mouse and keyboard are available to be used with the Pi.

WORKING WITH THE PiR2 CONTROLLER

USING THE PiR2 CONTROLLER

The PiR2 CONTROLLER has two components: the PiR2A electronic module and the PiR2 Software. In fact, the PiR2 Software will fully function without any PiR2 electronic module. This situation is said to be using a virtual PiR2 CONTROLLER. Any Raspberry Pi computer can optionally have a TV monitor plugged into its HDMI port. This permits the PiR2 CONTROLLER software to output video photos to the “imageTV0” ioDevice and output audio (sound messages) to the “audioTV0” ioDevice. By connecting a TV monitor to hdmi (which is hdmi0, the hdmi connector beside the power connector on the Pi 4B), these two ioDevices can be used by the PiR2 CONTROLLER Software. Simply put, a Raspberry Pi equipped only with a TV monitor can sense the processor temperature and send messages to the live operator of the PiR2 CONTROLLER Software. This means that any owner of a Raspberry Pi can install and run the PiR2 CONTROLLER Software. It also means that such an owner can also upload logs to the MehInCharge.com website. All that needs to be done is to communicate with the Webmaster (David@ColeCanada.com) and ask him to allow this additional Raspberry Pi to communicate with the MehInCharge.com website. To run the PiR2 CONTROLLER software, the owner of the Raspberry Pi will need a copy of the PiR2 software. As of 2020 D Apr 25, no-one has made such a request.

INSTALLING THE PiR2 CONTROLLER SOFTWARE

The PiR2 CONTROLLER SOFTWARE must be installed in a new folder named “PiR2” which has been created on the “Desktop” of the Raspberry Pi. A few other files are needed. They will be described in detail when the PiR2 CONTROLLER software is provided.

INVOKING THE PiR2 CONTROLLER SOFTWARE

To run the software, go into Terminal mode and type in “sh pir2.sh” at the command prompt. The PiR2 CONTROLLER software will start running and will ensure that no other instance of the PiR2 CONTROLLER software is running. (Two instances of the PiR2 software must NOT be running at the same time. The PiR2 software checks to ensure that another instance is not running before it starts.)

The PiR2 software will then examine the Raspberry Pi to see which PiR2 electronic hardware is available (if any). The software will then start running and will ask the live operator if a PiR2 command is to be run. The operator can enter one command after another, each followed by “Enter”. When finished the operator simply hits “Enter”. Examples of such commands are:

audioTV0:Coupe.mp3

This will play music (Little Deuce Coupe) on the TV speaker

acquire:tempProc

This will display the temperature of the Pi processor.

acquire:tempId00

This will display the temperature from the TMP102 sensor if the PiR2 is present

After hitting “Enter”, the PiR2 CONTROLLER software will begin sensing information every 10 seconds. The information will automatically be displayed and updated in the Terminal window. This information will also be written to a PiR2 log.

While the PiR2 CONTROLLER SOFTWARE is running, any other programs can be run on the Raspberry Pi. If another program changes a sensor or activates a control or changes an important disk file, it may interfere with the PiR2 CONTROLLER operation. But the operator should feel free to run any other programs, access the web etc. But the operator must NOT change any of the PiR2 software or data. To stop the PiR2 CONTROLLER SOFTWARE, always enter ^C (Ctrl-C) at the Terminal prompt. This will rename the log file correctly and will save it so that it can be synchronized with MehInCharge.com. If you simply turn off the power to the Raspberry Pi, then any recent log data will be lost.

SYNCHRONIZING THE LOGS

In order to synchronize the PiR2 CONTROLLER log with the MehInCharge.com server, the operator must use the Raspberry Pi to start browsing the web. Then the operator must go to the MehInCharge.com website. A novice operator should click on the “Log” button under “__VIEW__” to see the last record number that was previously uploaded to the MehInCharge.com website from this PiR2 CONTROLLER. To do this, scroll all the way down to the end of the log, which may take quite a while. (At time of writing, the latest record number in the author's log is 208846.) Then ask MehInCharge what it thinks is the latest record number on file. To do this click on the “View” button under “_FILES_”. It will show this same number to the left of the words “max Id / Cnt” above the “MIC files”. It is important to know the latest record number to prevent repeating the loading of logs that have already been uploaded. Do not be alarmed if the box to the left of “max rec Id” above “PiR2 files” is empty. This simply means that you have not started up MehInCharge.com from your Raspberry Pi or you have not installed the software. Remember . . . in order to upload files from the Raspberry Pi to MehInCharge.com, you must be logging into the web from the Raspberry Pi.

To actually synchronize log files, you must click on the “.Sync.” button under “_FILES_”. Then use the File Manager on the Raspberry Pi to first highlight, then drag and drop, the next log file onto the “Choose File” button. Then click on the “Sync Log button. This will cause the synchronization to begin. Once it is complete, you can then select the next log file to be uploaded (i.e. synchronized) from the Raspberry Pi.

After all synchronization is complete, any control files (PiR2 commands that were entered into MehInCharge.com for this Raspberry Pi) will be synchronized. This means that they will be downloaded to the Raspberry Pi and soon . . . automatically executed.

STARTING THE PiR2 CONTROL PANEL

The PiR2 CONTROL PANEL is in its infancy and does not provide much functionality. However it soon will be usable to send a control command to the PiR2 Controller. This must be done while the PiR2 CONTROLLER is running. It is the only way to invoke a control command without stopping (and restarting) the PiR2 CONTROLLER (or using MehInCharge.com as described earlier.)

First, the PiR2 CONTROL PANEL must be started up. To do this (as of 2020 D Apr 25):

- 1) On the Raspberry Pi Desktop, click the Raspberry icon (top left corner)
- 2) Click Programming
- 3) Click Python 3
- 4) Click Files
- 5) Click Recent Files (or the equivalent)
- 6) Click home/pi/Desktop/panelGUI.py
- 7) Click Run
- 8) Click Run Modules

The PiR2 CONTROL PANEL will appear in a Window. Data in it will not be updated. The main button that will soon function is the “SEND” button. Type an operator command in the box above it, then click on “SEND”. The command will eventually be performed.

-end of main document-

APPENDICES

**A) PARAMETERS WITH GPIO #s and PIN #s (as of 2020 D Apr 25)
(on the PiR2A prototype)**

| Parameter Name | SW Name | Global Pin Name | GPIO Pin # | Physical Pin # | Comment |
|----------------|---------|-----------------|------------|----------------|-------------------|
| ? | diCpin | g_diC_pin | na | na | not used |
| digIn0 | | g_di0_pin | 22 | 15 | =pushButton |
| digOut0 | do0pin | g_do0_pin | 17 | 11 | yellow LED (heat) |
| pushButton0 | | g_btn_pin | 22 | 15 | |
| redLED | ledRpin | g_ledR_pin | 27 | 13 | |
| whiteLED | ledWpin | g_ledW_pin | na | na | not used |

B) COMPLETE LIST OF ioDEVICES & PARAMETERS (as of 2020 D Apr 25)

To access the parameters, the following commands are typed as follows:

| Acquire-Type of Commands | Control-Type of Command |
|---------------------------|-------------------------|
| acquire:Parameter | Parameter:Operand |
| acquire:Parameter:Operand | |

Note 1. Many of these ioDevices and/or Parameters are not yet included in the software.

Note 2. the parameters shown in bold face below are implemented as of 2020 D Apr 25

Note 3. A virtual PiR2 CONTROLLER will not include Parameters of type E

Note 4. Parameters ending with a digit (0, 1, etc.) can apply to more than 1 ioDevice

Note 5. Most Acquire-type of commands do not have an operand

Note 6. The catBuffer can be cleared (:""), concatenated to or read.

Note 7. Every parameter that is read or written is always loaded into the catBuffer

Note 8. The emailWho for outgoing email defaults to the owner's email address

where A/B/C is Acquire/Control (B means **Both** Acquire and Control)

where D/E/I/O/P is Data/External/Internal/Optional/Personal types of parameters

| Parameter Name | A/C | D/E/ | IODEVICE | Operand |
|----------------|-----|-------|----------|---------|
| | /B | I/O/P | | |

| | | | | |
|-----------------|---|---|--|--------------------------|
| ambHeat | C | E | heatSource | :1 equivalent to digOut0 |
| ambLight | A | E | ambientLight | |
| ambTemp | A | E | ambientTemp | |
| ambHumid | A | E | ambientHumid | |
| areaCode | B | D | Code letter assigned to this piArea e.g. "A" | |
| audioIn | A | E | microPhone | \$datedAudio?.wav |
| audioJack | C | O | piAudioJack | :alarmA.wav |
| audioOut | C | E | PiR2 speaker | :alarmA.wav |
| audioTV0 | C | O | hdmi0 | :Coupe.mp3 |

| | | | |
|---------------------|-----|---------------|---|
| audioTV1 | C O | hdmi1 | :alarmA.wav |
| catBuffer | B D | catBuffer | :“Alarm High Temperature tempId00 is ” |
| ymdTime | A D | date/time | e.g. “2020 D Apr 25 05:42.157 GMT” |
| digIn0 | A E | | |
| digIn1 | A E | | |
| digIn2 | A E | | |
| digOut0 | C E | | :1 |
| digOut1 | C E | | :1 |
| digOut2 | C E | | :0 |
| doorbellPress | A E | doorbell | |
| emailAttachment1 | B D | email | |
| emailIn | B D | email | |
| emailOut | C D | email | |
| emailSubject | B D | email | |
| emailText | B D | email | :”doorbellPress=1 at ymdTime:“2020 D Apr 25 05:42.157 GMT” |
| emailTime | B D | email | |
| emailWho | B D | email | : to ” David@ColeCanada.com ” and from:”A_PiR2_in_621_Ottawa@MehInControl.com” or vice-versa |
| extWHum0 | A E | weatherSensor | |
| extWPress0 | A E | weatherSensor | |
| extWTemp0 | A E | weatherSensor | |
| imageCamera | A E | camera | \$datedImage?.gif |
| imageTV0 | C O | hdmi0 | :datedImage9.gif |
| imageTV1 | C O | hdmi1 | :logo1.gif |
| iRed | B E | infraRed | :1 |
| logA | B P | logA | :”any Text” |
| logB | B P | logB | |
| logC | B P | logC | |
| logLength | A D | | |
| moistSense | A E | | |
| motionSense | A E | | |
| ownerCommand | B P | command | imageCamera\$VisitorTest?.gif |
| piArea | B D | | |
| piSN | A D | | |
| piIP | A D | | |

```

procTemp    A  I
pushButton0 A  E
pushButton1  A  E

refreshDt    B  D
redLED      C  E           :1
taskSleepDt  A  D
taskSleepToday A D
taskSleepPC  A  D

tempId00    A  E
tempId01     A  E
tempId02     A  E
tempId10     A  E
tempId11     A  E
tempId12     A  E

thisHouse    B  D           :621_Ottawa
timeZone     B  D           :EDT
toOwner     C  P           :usb0_charger_is_empty.wav
unixTime     A  D           returns e.g. 0007865465789034
usb0         A  E
usb1         A  E
usb3         A  E

videoTV0     C  O  hdmi0    :tutorial01.mp4
videoTV1     C  O  hdmi1    :tutorial02.mp4
videoCamera  A  O           $datedVideo?.mp4
wavCreate    A  D           :tempAlarm.txt (see next section)
whiteLed     C  E           :0

```

Notes:

\$datedAudio?.wav assigns unixTime-stamp in ?, saves it and returns the full name

(This can be used to save a "Hey Google" audio wave.)

\$datedImage?.gif assigns unixTime-stamp in ?, saves it and returns the full name

\$datedVideo?.mp4 assigns unixTime-stamp in ?, saves it and returns the full name

\$datedImage9.gif discovers & uses operand where 9 is the most recent image

.gif is an image file

.mp3 is an audio file (often found on old CD sound tracks)

.m4a is an audio file (recorded using Windows-10)

.mp4 is a video (movie) file containing both audio and video information

.txt is a text file

.wav is an audio file (recorded using Linux or generated using espeak)

for logSetA type logA

C) Making An Audio Wav

The PiR2 CONTROLLER can audibly speak any text string, presuming it is in English for now. The wavCreate command uses existing espeak (see Source 08) software to create an audio file (in “.wav” format) from the given text file (in “.txt” format). This audio file sounds like a robot is speaking.

wavCreate reads pasteBuffer.txt, renames it to operand.txt, then makes a wav named operand.wav

The process to do this (using PiR2 commands) will be as follows:

```
catBuffer:
catBuffer:“Alarm Alarm Low Temperature tempId00 is ”
catBuffer:tempId00
wavCreate:alarmLow_tempId00.txt
audioTV0:alarmLowTemp_tempId00.wav
```

Explanation:

```
catBuffer: creates an empty pasteBuffer in catBuffer
catBuffer concatenates ”Alarm Alarm Low Temperature tempId00 is “ text string to catBuffer
catBuffer.txt will contain ”Alarm Alarm Low Temperature tempId00 is “
catBuffer concatenates the most recent value of tempId00 eg 29.99 to the catBuffer
catBuffer.txt will contain ”Alarm Alarm Low Temperature tempId00 is 29.99”
wavCreate will save the catBuffer.txt as alarmLow_tempId00.txt and then
(Note that the catBuffer remains unchanged.)
wavCreate will use espeak software to generate a wav named alarmLow_tempId00.wav
audioTV0 will send the audio wave named alarmLow_tempId00.wav to the TV speaker
the operator will immediately hear the PiR2 software speaking the alarm phrase through the TV
```

D) Handling of email by the PiR2 CONTROLLER Software

Incoming email

The PiR2 CONTROLLER is notified of an incoming email that is addressed to this PiR2 Controller by the emailIn flag; i.e. the emailIn is set to “1”. this means that all the email fields have been loaded with information in the email. The PiR2 should then

```
examine the contents of emailWho to read the sender's email address.
examine the contents of emailSubject to read the email subject
examine the contents of emailText to read the email text
examine the contents of emailTime to read the email time sent (in unixTime format)
examine the contents of emailAttachment1 to read the name of the email attachment #1.
```

The email is deleted when emailIn is set to “0” by PiR2. Even though the email is deleted the contents of all the email fields remain unchanged. This permits the PiR2 to respond to an email leaving some of the fields unchanged.

Outgoing email

The PiR2 CONTROLLER can send an email by loading the email fields with any new data that is desired. This is usually the way that the PiR2 CONTROLLER will notify an owner of an alarm situation. If no new data is loaded, the previous contents of each email field is used, resulting in an identical email being sent back to the sender, which usually achieves nothing.

The PiR2 CONTROLLER simply logs and doesn't reply to all email's received from any email address other than the emailOwner. All emails received from the emailOwner are also simply logged unless the emailSubject is "command". In this case the emailText is used as a PiR2 command (or series of commands). Some examples follow:

Example 1 acquire:procTemp

This would cause the PiR2 to read the procTemp. This command will be executed and will be logged. This does not result in an email reply being sent.

Example 2 catBuffer:
 catBuffer:"In my "
 catBuffer:piArea
 catBuffer:" at "
 catBuffer:ymdTime
 catBuffer:"the most recent procTemp is "
 catBuffer:procTemp
 emailText:catBuffer
 emailOut:1

All of these commands will be executed by the PiR2, one after another. The command on the last line (emailOut:1) will result in an email reply being sent. This is the way an owner of a PiR2 CONTROLLER can use email to send commands to the PiR2 and to receive an email reply containing information.

If only one PiR2 CONTROLLER is in use, the following simpler email command request can be used:

Example 3 emailText:procTemp
 emailOut:1

This will result in only the most recent value of the procTemp being sent as the reply.

E) OMNI-PURPOSE HIGH/LOW SENSOR CONDITIONER CIRCUIT

Any high/low input sensor is connected to either

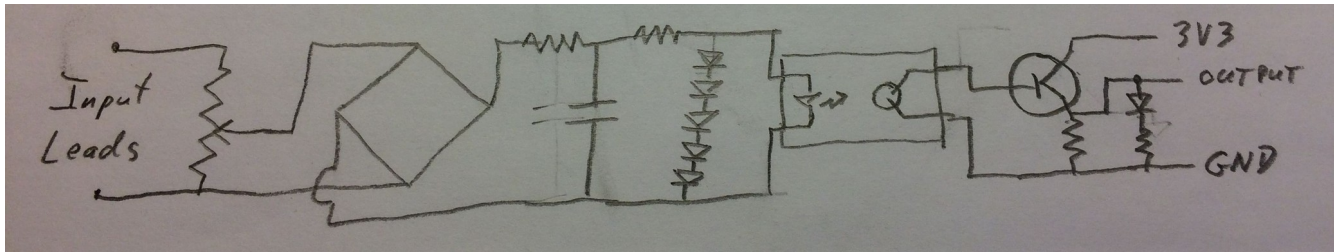
- 1) a normally-open switch like a reed switch with no power source . . . or to
- 2) some source of power that is either a DC or an AC voltage.

1) A normally open switch should be read as 1 (3.3v) when open and 0 (0v) when closed. The circuit for this is 3v3—10k—Output ----reed-leads----1k--- GND. Fortunately this circuit does not have an exposed 3v3 lead which could seriously hurt the power supply if it is accidentally grounded.

2) The second case (a powered input) usually requires a different circuit for the cases of DC or AC voltages and for different polarities of the DC voltages. Furthermore, any direct connection exposes the sensor (and its computer) to damage due to possible accidental exposure to the wrong type or high value of voltage.

An omni-purpose conditioner circuit should separate the powered input from the GPIO sensor of the Raspberry Pi. The circuit described below provides a high degree of protection for the sensor circuitry, including complete isolation between the circuit being sensed and the computer.

An omni-purpose input sensor conditioner circuit is shown below:



input – potentiometer – full-wave rectifier – filter – voltage limiter – opto-isolator – transistor - LED - output

The values of the components will be added at a later date.

The simple steps to set it up (to attach the input and output and do the adjustments) are:

- Turn the potentiometer fully counter clockwise (lowest voltage).
- Turn off the input source (e.g. doorbell not pressed).
- Connect the output to the GPIO sensor of the Raspberry Pi.
The LED should be off. If it isn't, short the input leads together (temporarily).
If the LED is still on, speak to the provider of the circuit.
- Connect the input source.
- Turn on the input (e.g. doorbell pressed).
If the LED lights up, go to step g). If the LED does not light up . . .
- Turn the potentiometer clockwise until the LED lights up, then turn it 10% more.
- Turn off the input source (e.g. doorbell not pressed). The LED should turn off.
If the LED turns off, the input adjustment is complete.

This conditioner circuit is applicable to many different input sources such as:

- doorbell
- furnace thermostat
- etc

F) EXTERNAL BOOKS AND SOURCES

- Source 01 <https://www.tomsguide.com/us/alexa-vs-siri-vs-google,review-4772.html> Smart Home
- Source 02 <http://ephotocaption.com/a/60/60.html> Raspberry Electronics Projects
- Source 03 <http://ephotocaption.com/a/130/130.html> Electronics for the Raspberry Pi
- Source 04 <http://www.ti.com/lit/an/slva704/slva704.pdf> Understanding the I2C Bus by TI
- Source 05 <http://www.mehincharge.com/> www.MehInCharge.com Site
- Source 06 <http://ephotocaption.com/a/128/128.html> MIC and the PiR2 Controller
- Source 07 <http://ephotocaption.com/a/132/132.html> Python Programming Language
- Source 08 <http://espeak.sourceforge.net/> Brief description of espeak

/PIR2_CONTR_FUNC_08.odt as of 2020 D Apr 27

-end-