Appendix III
Definitions – Phase 2

Raspberry Pi Side/Qt

/etc/lirc/lircd.conf – Remote IR mapping (currently at 38kHz)
/home/pi/modules/test2/test2.pro – Qt project initialization
/home/pi/modules/test2/mainapp.h – MainApp class definition
/home/pi/modules/test2/i2c.h – I2C class definition
/home/pi/modules/test2/main.c – QCoreApplication loader
/home/pi/modules/test2/mainapp.c – main system for mqtt and lirc

- void messageArrived(struct mosquitto *m, void *obj, const struct mosquitto_message *message) [for testing]
  - convert message payload into QByteArray, parse and then classify message received by mosquitto subscriber
- MainApp::MainApp()
  - mosquitto initialization and loop initialization
- MainApp::~MainApp()
  - mosquitto loop stop, destroy and library cleanup
- MainApp::timerEvent(QTimerEvent*) / every 1 second
  - get sensor readings
  - convert message payload into QByteArray, parse and then classify message received by mosquitto subscriber
  - message format from server to device: *SS;ON/OFF/r;[temperature]
    - ON: turn AC on or change temperature
    - OFF: turn AC off
    - r: read sensor data
  - For each type of command,
    1. Build LIRC response
       execute system call of LIRC
       format: irsend SEND_ONCE MY_REMOTE COMMAND
       COMMAND: a proper and suitable keypress defined in lircd.conf
    2. (If command is r) read sensor data
    3. Build MQTT response

format: DD;P;T;S;C where P = power (ON/OFF), T = temperature (float), S = set/target temperature (int), C = current (float)

/home/pi/modules/test2/i2c.cpp – I2C object and methods to get data/command an I2C device

I2C::I2C()

initialization for I2C device

float I2C::getTemperature()

assign 0x48 to the I2C port

sends command 0xF3 (temperature)

returns temperature in °C

float I2C::getCurrent()

initialize I2C bus and device (location 0x49)

commanding the device by sending configurations:

0x01 – select config register

0x80 - AIN0 and AINN = AIN1, +/- 6.144V

0xE3 - Continuous conversion mode, 860 SPS

read 2 bytes (msb, lsb) reading

conversion from binary to voltage, and then voltage to current

returns current level

Server Side (QtWebApp outside Raspberry Pi –works if and only if the MainApp in Raspberry Pi is running)

MqttServer/MqttServer.pro – Qt project file configuration

MqttServer/MainServer.h – server definition

MqttServer/main.cpp

get .ini configuration file, load settings for web app, initiate main application, session setting, HTTP listener setting

MqttServer/MainServer.c

generating HTML page to control the system

void messageArrived(struct mosquitto *m, void *obj, const struct mosquitto_message *message)

Getting acknowledgements and sensor data from Raspberry Pi, then updating the shown HTML values for each data

Format: DD;P;T;Pr;C where

P = Power (ON/OFF)

T = Temperature from sensor
Pr = Preset temperature
C = Current reading from sensor

MainServer::MainServer(QObject* parent) : HttpRequestHandler(parent)
creation of main server, setting timers for routine sensor reading

void MainServer::mosquittoInit()
mosquitto initialization

void MainServer::timerEvent(QTimerEvent* ev)
send reading sensor command every 1 second

void MainServer::service (HttpRequest &request, HttpResponse &response)
HTML page building and button response processing

How to Start the System

1. Run the main system in Raspberry Pi [can be automated]
2. Run the server, it will establish the connection and posting the html page to the host and port specified in the .ini file
3. Open host:port in the browser, the control and monitor panel is ready to use.

Hardware Calibrations

Celsius = (output)(175.72)/65536 – 46.85
Volt = (output)(5.14)/27468
Ampere = (volt)(625/3000) + 0.1092