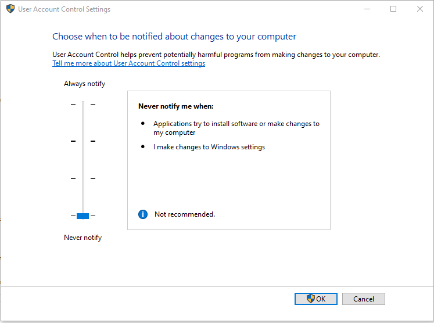
MyPAM Middleware Documentation

# Important General information

## UDP Communication Datagram

* The data between modules is sent in this format: “Sending\_module\_name|Type\_of\_Data\_contained\_or\_request|data” Example: “GM|gameTypes|CP-P2P-Other” - It is from the game’s module (GM), it contains the available game types and the data it wants to send is “CP-P2P-Other”.
* When sent to any module, the string is split at every “|” and then sorted by the “listen () function in each module.

## Testing

* Make sure to run the installer before trying to test the code through visual studio as part of the code require files withing the installer.
* Also, you will be prompted before running but run visual studio as administrator.
* To test all the code in visual studio run it from the “Start MyPAM Midwear” program.
* The “Central\_UDP\_Server”, “Main\_Menu” and Start MyPAM Midwear module all have a Boolean variable at the top of their code called “releaseMode” this is used for testing purposes so the modules know where to run other modules and programs from. Set it to true when creating an installer file so the programs know to run other modules from programData and programFiles on the users PC. Set it to false when compiling in visual studio so the program knows to find the programs in their respective compile locations.
* Also, when setting up the middleware on a PC please ensure you turn down its “User Account Control Settings” to never notify so the middleware can run automatically with no human interaction.

## Compiling

* When compiling each module highlight can compile each module excluding the installer as the installer takes a while to compile.
* Only compile the installer when needed to save time.

## General Functions in All Modules

### listen()

* This function’s roll is to sort the incoming data.
* First the data is split. Then the sorted into where it is from and what it is requesting/ contains so the program knows what module to send the data too.

### receiveData()

* Here is where the module waits to receive incoming transmission from other modules.
* There is a try() catch() encase of unforeseen problems just so the module doesn’t crash entirely.
* This is also used in some modules to enable connection checking with timeouts.

# Central\_UDP\_Server

## Brief

The purpose of this module is to pass data between modules. This makes it easy to send copies of data to multiple modules at once as well as making it easy to remove and add new modules.

## Important Information

UDP listen port: 4000

## Functions

### Main()

* The main function is used to setup the module when it is first run.
* The console is hidden from the user.
* Then UDP communication is set up.
* Lastly the module listens for incoming communications.

### UDPCommsSetup()

* This function’s sets up the parameters for the UDP listener.
* Most of the code in this module is self-explanatory.

### UDPsend(string datagram, int port)

* Here the data being sent and what port it is being sent to is passed in and the function sends the data to the desired port.

# LLC\_Communication

## Brief

This module takes data from the lower level controller (LLC) and passes it to the central server and vice versa. The module is also used to check the connection status of the LLC and edit crucial data from the games before reaching the LLC.

## Important Information

UDP listen port: 60000 (Comms from LLC)

UDP listen port: 3200 (Comms from Central Server)

## Functions

### Main(string[] args)

* The main function is used to setup the module when it is first run.
* The console is hidden from the user.
* Then UDP communication is set up.
* The module then starts its communication start up procedure where it tells the safety model it is ready to run and waits for a response back telling it to start.
* The timeout is set to 50ms, after 50ms of not receiving anything the module will try again 15 times before deciding something has gone wrong

### commsSetup()

* Sets up two listeners and endpoints for communication between the module and the central server and the module and the LLC.
* Buffers are set to zero to remove the effects of bottlenecking at different locations.

### UDPsendToServer(string datagram)

* This is used to send data passed into the function to the server.

### UDPsendToController(string datagram)

* This is used to send data passed into the function to the LLC.

### beginControllerGameComms()

* This function is the main communication protocol while the user is playing games.
* The program will wait to receive data from the LLC then read that data and pass it to the games.
* Then the module will wait to receive data from the central server (games) and then edit that data before sending to the LLC.

### editGameData(string gameData)

* Here the game data is edited with data provided by the database about the users as well as changing the error checking character (A/B).

### readMyPAMData(string controllerData)

* In this function the new data received from the LLC is read and the connectionChecking parts are saved in the program.

## Classes

### gameDataFormat

* This is the format of the JSON file coming from the games.

### controllerData

* This is the format of the JSON file coming from the controller.

# Games\_Module

## Brief

This module takes data from the games and passes it to the central server and vice versa as well as preforming all other task that are to do with the games (e.g. adding, removing…). It will also check the connection to the games to make sure they haven’t crashed and act accordingly if they have.

## Important Information

UDP listen port: 2300 (Comms from Game)

UDP listen port: 2200 (Comms from Central Server)

All games have their listen port set to 3000.

## Functions

### Main()

* The main function is used to setup the module when it is first run.
* The console is hidden from the user.
* Then UDP communication is set up.
* Then the module will listen for incoming data from other modules

### UDPCommsSetup()

* Two listeners and endpoints are set up for communication between the games and the module and the central server and the module.
* The buffers are set to zero to remove the effects of bottlenecking.

### UDPsend(string datagram)

* This function is used to send data to the central server

### startGame(string type, string name)

* This function is passed the name of the game and its type so that it can locate and load the desired game for the user.
* Its will then inform the safety module that the game has started.

### gameTypes()

* This function returns the current list of game types added to the middleware.

### gameNames(string cBox1Text)

* This function returns the list of games available to be played.

### removeGame(string cBox1Text, string cBox2Text)

* This function is provided the type of game and game name so it can them remove that game from the middleware.

### addGameType(string tBoxText)

* This function is provided a game type name it will then create a new game type folder with this name.

### addGame(string fileName, string cBox1Text)

* This function is provided the location of the .exe of the game you wish to add to the middleware.
* Then it moves that game into its games folder saved in program Data so that its can be played by the user.

### setControllerType(string controllerType)

* The user will select the controller type in the settings menu, their choice is then saved in the settings file by this model.

### setMode(string mode)

* Like the controller type the user will choose what mode the middleware is in (either in install or patient mode). This is sent to the games module and then saved into the settings file to remember the user’s choice.

### storedSettings()

* Gets the current settings list and returns them as an array.

### storedSettingsString()

* Gets the current settings list and returns them as a string that can be sent over via UDP.

### UDPsendInGame(string datagram, int port)

* This function is used to send data to the central server or games, depending on the port passed into the module, when games are being played.

### controllerGameComms()

* This function is the protocol for when the user is playing a game.
* First it waits for dada from the central server (LLC) and send that to the games
* Then it waits to receive data from the games and send it to the central server when received.
* Connection timeouts are set to 50ms either way.

# Main\_Menu

## Brief

## Important Information

UDP listen port: 1400

## Functions

### mainMenu()

* The main function is used to setup the module when it is first run.
* Then UDP communication is set up.
* Then then it will see if the database module is running and if it is will shut it down.
* Then it will run all the necessary modules and initialised it UI components.
* Then it will request the game type installed on the middleware.
* Finally, it will check to see if the middleware is in patient mod and if it is will start the game selection menu automatically.

### UDPSetup()

* This module sets up the UDP listener on port 1400.

### startGameSelectionMenu()

* This function runs the game selection menu.
* Then its waits to receive information form the game selection menu telling it to resume or to start one of the available games.

### button2\_Click(object sender, EventArgs e)

* This checks to see if the settings button is clicked on.
* If it is then the setting menu will pop up.

### button3\_Click(object sender, EventArgs e)

* Checks to see if the start game button is clicked and if it is will run the games selection menu.

### startGame(string gameName, string gameType)

* First the module checks to see if any of the controller communication module are running or if the user selected game is already running and shuts them down.
* Its then runs the safety module.
* Then send a request to start the game.
* It then waits to receive confirmation that it can start running again and resets itself for the user.

### button4\_Click(object sender, EventArgs e)

* Check to see if the add game button has been clicked.
* If the user has selected a game type for the game they will then be prompted with a window where they must find and select the .exe for the game they wish to add.
* The information about the location of that game is then passed to the game module where it adds the game to the middleware.

### Button6\_Click(object sender, EventArgs e)

* This refreshes the information stored in the game type drop down.

### button5\_Click(object sender, EventArgs e)

* Check to see if the remove game has been clicked.
* Then make sure a game and game type has been selected before requesting the chosen game to be removed by the game module.

### refershGames()

* This function gets the games for the requested game type and stores them in the drop down.

### UDPsend(string datagram)

* This function will take the string passed to it, add a header and send it via UDP to the central server.

### UDPsendToSelectorMenu(string datagram)

* This function will take the string passed to it, add a header and send it via UDP to the game selection menu.

### closeAll()

* Checks all the possible modules that could be open and closes them.

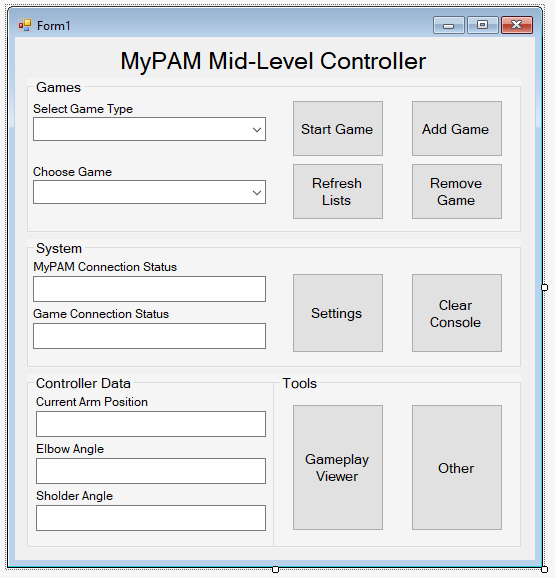
### openAll()

* Opens all the required start up modules

### Exit(object sender, EventArgs e)

* Attempts to close all modules and sends a UDP signal telling all modules to close themselves (Doesn’t work reliably).

## Interface



# Start MyPAM Midwear

## Brief

This module is used to make sure all the other modules are closed before reopening them to start a fresh. This is import because only one UDP listener can be open with the same port at one time, so it is important that you don’t try and open one of the modules when it is already open because both module will crash stopping the middleware from functioning properly

## Important Information

Must be run as administrator and will run as admin by default.

When setting up the middleware on a PC please ensure you turn down its “User Account Control Settings” to never notify so the middleware can run automatically with no human interaction

## Functions

### Main(string[] args)

* First the console is hidden from the user.
* Then it closes al the already open modules.
* It then check to see if all the modules are actually closed, if not it tries again.
* Then once they are all closed it will run the main menu and close itself.

### closeAll()

* Checks to see if all the modules are closed and if not closes them.

# Database

## Brief

This module builds and maintains an SQlite local database that countians data about the user and their usage of the MyPAM.

## Important Information

UDP listen port: 41200

## Functions

### Main(string[] args)

* The console is hidden from the user.
* Lists to hold the games played in this session, play times of each game and a list to hold the users controller and game data while they play the games.
* The UDP listener is setup.
* It then check to see if a database exists and if not it creates one with the tables determined at the top of the code.
* Some basic user information is then taken and added to the database.
* The module is then told to listen for incoming UDP transmissions.

### listenerSetup()

* Sets up the UDP listener on port 41200.

### addUser(string username, string password, string faceID) (Function in development apart of login)

* This function adds a new user to the database.

### createSessionData()

* Creates data about the users current session and adds it to the database.

### createGameSessionData(string gName)

* Creates data about the users current gaming session and adds it to the database.

### updateUserInfo()

* Updates the current user information with new data.

### updateSessionData()

* Updates the current session information with new data.

### updateGameSessionData()

* Updates the current gaming session with new data.

### addToGameplayList()

* Adds new gaming data to the gaming data list.

### outGameplayDataToDatabase()

* When the user closes their games this function output all the data in the “gameplayDataList” list into the database.

### findUser() (Function in development apart of login)

* Searches for users in the database an returns if the exits or not.

### UDPsend(string datagram)

* Takes the data passed into it, attaches a header and send it to the central server.

### closeDatabase()

* Calculates the most played game and the total session time and adds it to the database.
* Closes the database.
* Closes the program.

# Log\_In\_Menu (In Development)

## Brief

Used to login users into the MyPAM.

## Important Information

UDP listen port: 50100

## Functions

### LogInBut\_Click(object sender, EventArgs e)

* If the login button is pressed it takes that information inputted from the user and sends it to the central server to check if the user’s information exists.

### UDPsend(string datagram)

* Takes the data passed into it, attaches a header and send it to the central server.

# Safety\_Module

## Brief

The safety module is used to safely start-up the game – middleware – lower level controller communication and acts when one of the communication links is lost.

## Important Information

UDP listen port: 1600.

## Functions

### Main(string[] args)

* This function hides the console from the user.
* Then sets up the UDP listener.
* Then waits until both the game and the controller communicator modules have confirmed they are running.
* Once they are a set up sequence begins by sending “startGameListen” to the central server.

### commsSetup()

* Sets up a UDP listener on port 1600.

### UDPsend(string datagram)

* Takes the data passed into it, attaches a header and send it to the central server.

### runConnectionLostProtocol(string from)

* When a connection loss is detected this function runs.
* Currently it attempts to close all controller modules.
* Then tells the main menu to resume.
* The closes itself.

# Settings\_Menu

## Brief

This module allows the user to change what controller module to uses, set if the middleware is in patient mod or not and add new gameTypes folders.

## Important Information

## Functions

### settings()

* First the module check what controller type has been previously selected and saved by the user and displays this.
* And checks if the module is in patient mode or not again displaying it in the correct check box.

### button1\_Click(object sender, EventArgs e)

* Send the game type the user wished to add to the central server if “Add Type” is clicked.

### checkBox3\_CheckedChanged(object sender, EventArgs e)

* Checks if check box 3 has checked if it has it sets the controller type to LLC in the settings text file.
* If it has been unchecked it runs the “noControllerSelectedCheck” function.

### checkBox1\_CheckedChanged(object sender, EventArgs e)

* Checks if check box 1 has checked if it has it sets the controller type to VC in the settings text file.
* If it has been unchecked it runs the “noControllerSelectedCheck” function.

### checkBox2\_CheckedChanged(object sender, EventArgs e)

* Checks if check box 2 has checked if it has it sets the controller type to MiniMyPAM in the settings text file. (May be removed as an option)
* If it has been unchecked it runs the “noControllerSelectedCheck” function.

### noControllerSelectedCheck()

* If no controller has been set then the LLC (lower level controller) will be set to the chosen controller type by default.

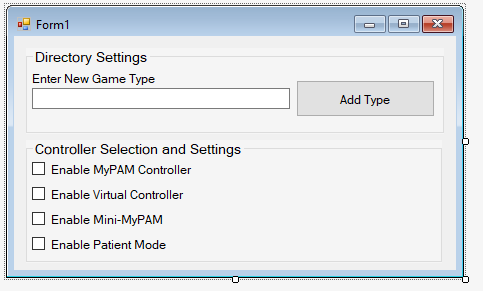
### UDPsend(string datagram)

* Takes the data passed into it, attaches a header and send it to the central server.

### CheckBox4\_CheckedChanged(object sender, EventArgs e)

* Checks if the user has set the middleware to be in patient mode or not and saves it in the setting text file.

## Interface



# Virtual Controller Module (Still works but need to be updated)

## Brief

Fundamental the same as the LLC module with some small changes that allow it to work better with the virtual controller.

## Important Information

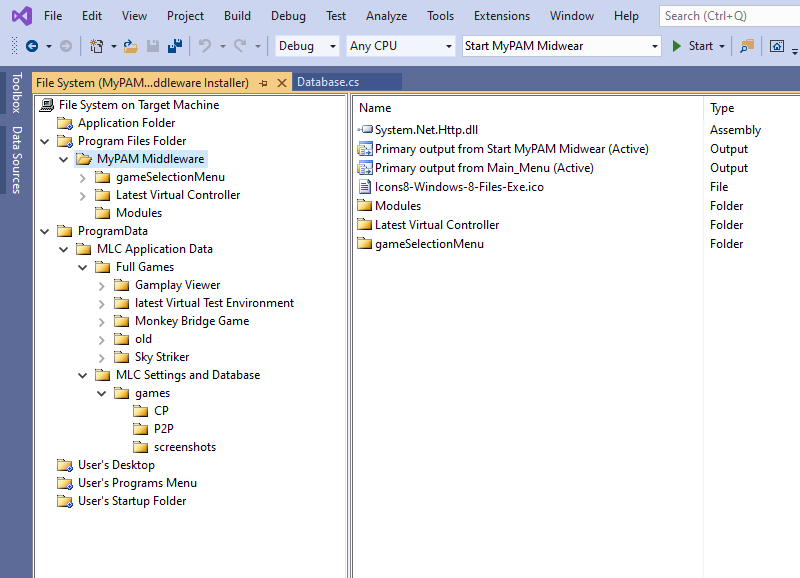
UDP listen port: 2600 (Comms from VC)

UDP listen port: 2500 (Comms from Central Server)

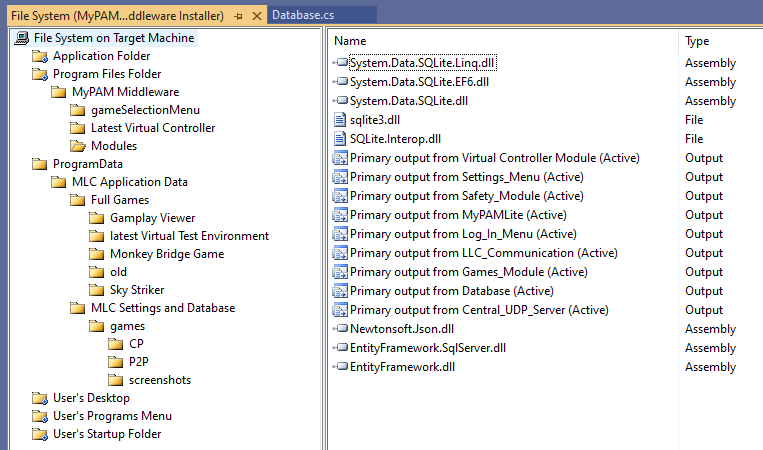
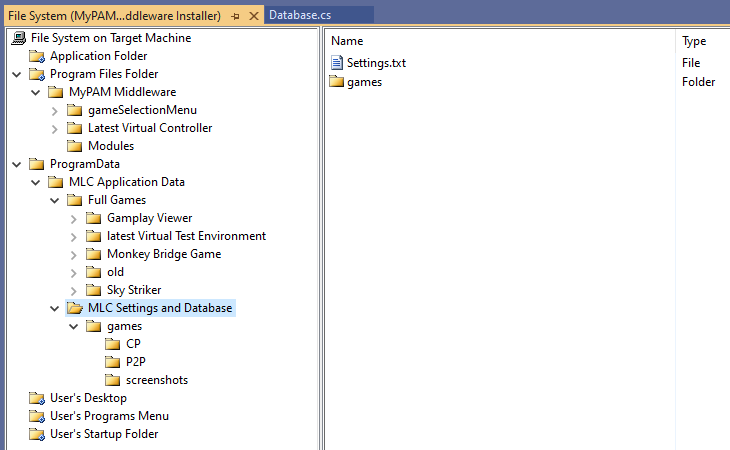
Virtual controller port is 3500

## Functions

# MyPAM Middleware Installer



* All files that are required to be preinstalled can be found in “Default\_Programs” folder of this project.
* All path must have the same names as show for the middleware to function if the installer is remade.



* Must include a “Settings .txt file”