TABLE OF CONTENTS

INTRODUCTION

EasyGUI is an easy and simple way to create a Graphical User Interface on an Android device without any Android programming experience.

The system works by sending simple messages via Wi-Fi (using TCP) from any device and any operating system, to a target Android device (Mobile Phone or Tablet). These messages are simple strings that allow the creation of 12 different Widgets and a virtually

The most significant thing about these Widgets are that they are dynamic. Meaning that their size, shape, position, and color can be changed even after they were created, even during run-time.

Basically EasyGUI works as a Client, communicating with a Server on another device. The server may be ANY kind of hardware using ANY Operating System. All the logic resides in the Server which can be written in ANY language, (Python, Basic, C, C++, Pascal, JavaScript, Lua etc.) as long as they can send and receive messages using Sockets. This way most of the effort in creating an App can be concentrated at the “Business End” in the Server.

DEFINITIONS

Client: An Android device, a Mobile Phone, an Android Tablet or a Laptop running the Android Operating System

Server: Any kind of Computing device, A Desktop, Laptop, Mobile phone, Single Board Computer (SBC) that is capable of Wi-Fi communication using Sockets.

It may be an Apple Phone or Tablet, An IBM compatible desktop, a Linux system, Another Android device of any kind, or an SBC like the ESP32 or an Arduino.

Template: One or more pages containing EasyGUI widgets.

Widget: A Graphical User Interface Item, that allows communication with the user.

COMMAND FORMAT

Each Command consists of a Key/Value pair, where the Key is separated from the Value by an Equal (=) sign. Ie. 01=MyButton

Multiple Commands can be strung together where each is separated by a Tilde (~).
Ie. 01=MyButton~Left=500~Top=300~Text=Press This

The first Command in a multiple Command line must be the command that identifies the Type and ID of the Widget.

The ID of a Widget must be unique. Even in a multiple Page Template.

All keywords are Case sensitive. There must NOT be spaces between separators.

In a Multi Page Template, The first page is always "Page0, followed by Page1, Page2. etc. in that strict sequence. There are no practical limits to the number of pages a Template may contain.

All colors are expressed as 9 digits RGB

CLIENT SERVER COMMUNICATION ↑

The Server will be loaded first, and will display it's IP address and Port Number. waiting for the Client to initiate the conversation.

The first time the App is loaded it will prompt you for the last three digits of the Server's IP (leading zero fill to 3 digits) and the Port no (leading zero fill to 5 digits).

After that the Client will store this data, and it will only be necessary to re-input it if the Server's IP address or the Port number has changed.

Example: if your Servers IP address is: 192.168.24.73 and the Port is 4000, type in the following when prompted: 07304000

The Client will send a start message first, containing it's IP and other identifying information.

After The initial handshaking, ALL TRANSACTIONS MUST BE IN PAIRS. Ie. Each **TRANSMIT (Put)** must be followed by a **RECEIVE (Get)**

The Server will send Commands to the Client to create or change the Widgets. The Client will reply to each message with either: OK (Message 40) or ERROR (Message 96), or in the case of a SYNTAX ERROR 99.

It is the Servers responsibility to check if an error has occurred.

A typical sequence of events:

1. Create a Template, either using EasyGen or manually using SimpServer.py
2. Server issues a Wait (40) command to wait for user action
3. The User performs an action (like pressing a Button)
4. The Client replies passing 3 or pieces of information
    a. The type of the Widget
    b. The ID of the Widget
    c. The action the user performed (like short press/long press)
5. The Server takes the necessary action based upon the Reply from the Client
6. Repeat from Step 2

COMMANDS

These Commands are contained in the supplied EasyGUI.py module. These are provided to make the programming of the Server even easier.

It's use is not compulsory or necessary in order to use EasyGUI, you can do all the detailed server programming yourself.

Open (Port) - This will open communication on the Port number specified.

Put (message) - Sends a message to the Client to create or change a widget. All the Widgets parameters will be updated in the Dictionaries in the config.py module which holds the current parameters of all widgets.

Get () - Receives a message from the Client, and returns the Client's reply

Send(message) - The combination or Put and Get. Returns the Client's reply

Find (ID,Key) - Returns the current value of the Widget's as per the Key.

if the following message was sent:

1=MyButton~Left=500~Top=300~Color=255000000

Find('MyButton','Top') will return 300

Wait() - Sends a message **40** to the Client, and waits until a user action, which will be passed to the Server by the Client. The format of the return message will be

40=Button-1~2 (A long press of Button-1)

GetTemplate(FileName) - Reads a Template, which is stored on the Client and displays it. It will also update the dictionaries in config.py which holds the current values of all widgets.

EXAMPLES ↑

Python example where you can type each command manually, and the widgets will be created/changed interactively.

```
# SimpleServer.py (Using EasyGUI - No error checking performed)
EasyGUI import *
import config from EasyGUI import *

Open(4000)                              # Open Socket
while True:
        Mess = input('Send: ')          # Type in Command
        Put(Mess)                       # Transmit Command
        Print(Get())                    # print the received reply
```

This Garage Door Opener Server demonstrates the power of EasyGUI. It uses 2 Image widgets (the Garage and the Door) and 1 button. The Template only contains 4 Commands.

```
# Garage.py (Using EasyGUI)
import time from EasyGUI import *
import config

def GoUp():                             # Function to make the door go up
        for n in range(400,0,-10):
        time.sleep(.1)
        Send('11=door~Height=' + str(n))
        Send('01=up~Text=DOWN')
def GoDown():                           # Function to make the door go Down
        for n in range(0,480, +10):
        time.sleep(.1)
        Send('11=door~Height=' + str(n))
        Send('01=up~Text=UP')
Open(4000)                              # Open Socket
Send('51=Load~Text=Garage.dat')         # Load the template from the Client
time.sleep(.5)                      # sleep for half second
while True:
        Wait()                          # Wait for button press
        GoUp()
        Wait()                          # Wait for button press
        GoDown()
```

This is an example of a server using Micropython for Single Board Computers (like 8266 or ESP32).

```
#MicroPython SimpServer (Without EasyGUI and with Error checking)
import sys
import socket

CRLF = '\r\n'
port = 4000
s = socket;

def Put(mess):
         c.send((mess + CRLF).encode())     # Send message
def Get():
        return c.recv(1024).decode()        # Read reply
def Open(port):
        global c
```

```
        global s
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind( ('', port) )
        s.listen(10)
        print ('Listening on Port ' + str(port))
        c, a = s.accept()
        print ('Client connected', a)
        print(Get())
    def Close():                            # Close Socket
        c.close()
        s.close()
    def Start():
        Open(port)                          # Open Socket
        Put('51=Load~Text=Sample.dat')      #Load from Client
        print(Get())

    Start()
    while True:
        Put('40')                                   # wait for Client's reply
        rec = Get()
        if (rec == '40=Radio~2'):                   # Finish, close
            Put('Exit')
            Close()
            print('Socket Closed')
            sys.exit()
        if (rec == ''):                             # Client offline
            Close()                                 # Close socket
            print('Reopening')
            Start()                                 # Reopen Socket
            print(rec)
```

This an example Server written in JavaScript:

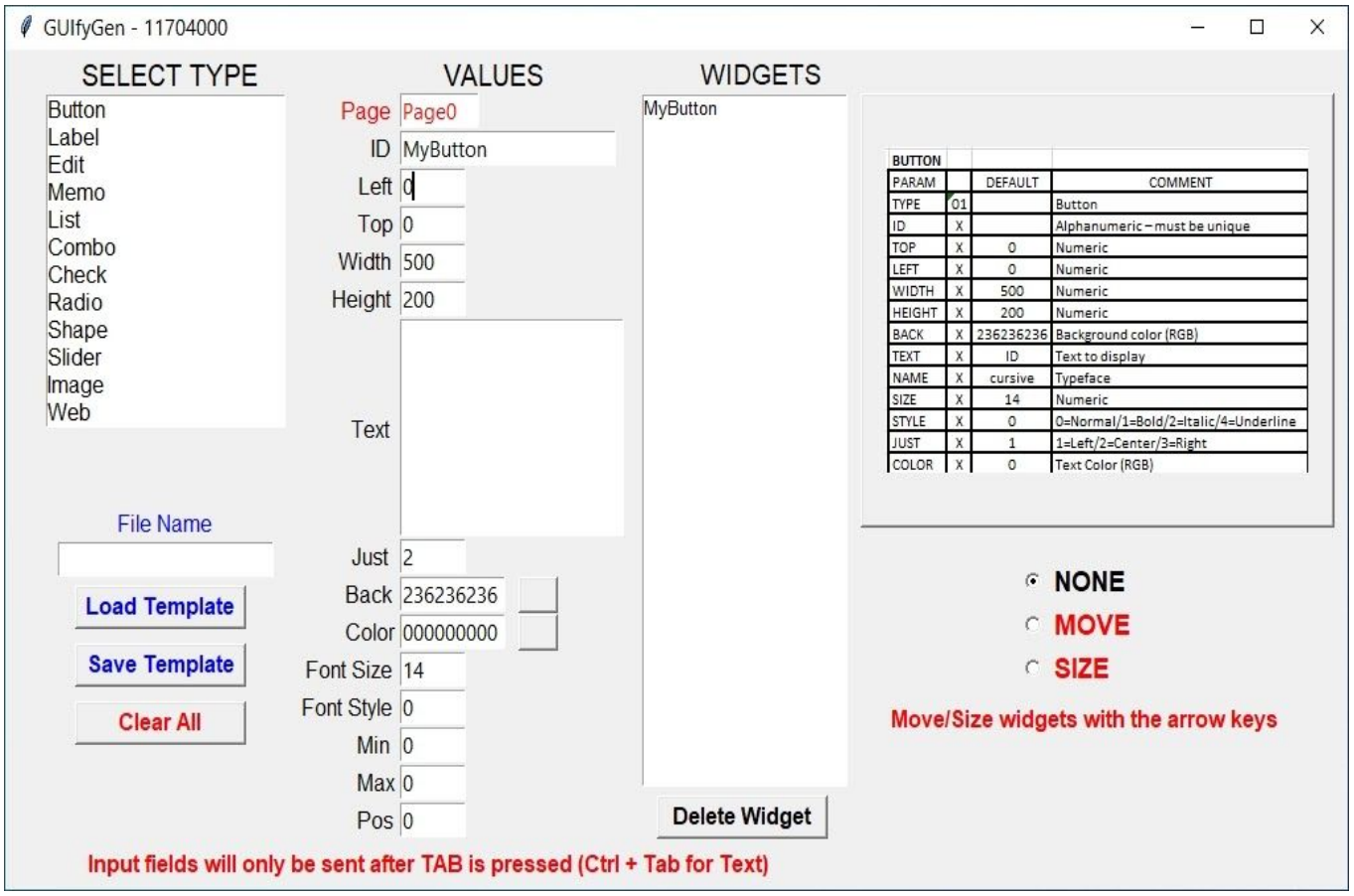
```
const net = require('net');
const server = net.createServer((socket) => {
  console.log('Connection from', socket.remoteAddress, 'port', socket.remotePort);

  socket.on('data', (buffer) => {
    //console.log('Request from', socket.remoteAddress, 'port', socket.remotePort);
    console.log('Recieved:',`${buffer.toString('utf-8')}\n`);
    var msg=require('prompt');
    msg.start();

    msg.get(['command'], function (err, result) {
    //
    // Log the results.
    //
    //console.log('Command-line input received:');
    console.log('  command: ' + result.command);
    socket.write(`${result.command}\n`);
  });
  });
  socket.on('Exit', () => {
    console.log('Closed', socket.remoteAddress, 'port', socket.remotePort);
  });
});
//server.maxConnections = 20;
server.listen(4002);
```

An Example for the 8266 using Arduino code:

```
#include "ESP8266WiFi.h"

const char* ssid = "yourNetworkName";
const char* password =  "yourNetworkPass";

WiFiServer wifiServer(80);

void setup() {
  Serial.begin(115200);
  delay(1000);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting..");
  }
  Serial.print("Connected to WiFi. IP:");
  Serial.println(WiFi.localIP());
  wifiServer.begin();
}

void loop() {
  WiFiClient client = wifiServer.available();
  if (client) {
    while (client.connected()) {
      while (client.available()>0) {
        char c = client.read();
        Serial.write(c);
      }
      delay(10);
    }
    client.stop();
    Serial.println("Client disconnected");
  }
}
```

THE GENERATOR

In order to make it easier to design Widgets, and place them on the Page, we have developed the EasyGen Generator. This Graphical User Interface Designer will allow the designer to see the end result on the Client device itself as the design is being developed.



How to use EasyGen:

1 - Start EasyGen (python EasyGen.py)

2 - Load the App on the Client. The EasyGen GUI will be displayed

Select a Widget from the Select Type list by clicking on it

a - Page0 will be the default

b - Type in a unique ID. The Widget will appear immediately on the Client's screen in position x=0, y=0, and it will appear in the Widget List

c - If you want to move or resize the Widgets, click on the Move or Size button and use the 4 arrow keys to move and or resize the Widget. Each press of the arrow keys will move or resize the widget by 10 pixels.
As the arrow keys manipulate the Widget the fields on the Gui will change automatically.

d - Fill in the rest of the fields, as necessary

e - Add more widgets

3 - If you want to edit any of the Widgets, click on it in the Widgets list, and all it's parameters will, be displayed so you can edit them

4 - When the Template is Completed, type in the Name of the file that you want to save it to on the Client and press 'Save Template'

5 - If you want to edit an existing Template, Type the name of the file into the FileName entry and press 'Load Template'

Since this Script is provided in Source format, you are welcome to modify it, after which any responsibility for the results will be yours .

WIDGETS

BUTTON

PARAM	TYPE	DEFAULT	COMMENT
Type	**01**		**Button**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		200	Numeric
Text			Displayed Text - if blank, ID is used
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~1 for short press, 2 for long press

LABEL

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**02**		**Label**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		200	Numeric
Text			Displayed Text - if blank, ID is used
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~1 for short press, 2 for long press

EDIT

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**03**		**Edit**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		150	Numeric
Text			Default Text to display - * if password
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~My Name (The entered line)

MEMO

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**04**		**Memo**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		500	Numeric
Text			Displayed Text
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~Line1|Line2 (Lines entered separated by “|”

LIST

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**05**		**List**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		500	Numeric
Text			Displayed Text, Lines separated by “|”
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~Line2 (the selected line)

COMBO

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**06**		**Combo**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		150	Numeric
Text			Displayed Text, Lines separated by “|”
Just		1	Text Justification - Left = 1, Center = 2, Right = 3
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~Item3 (the selected Item)

CHECK

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**07**		**Check**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		150	Numeric
Height		150	Numeric
Text			Text to display with the Checkbox
Just		1	Display Text at Left = 1, Right = 2, Top = 3, Bottom = 4
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min			Not used
Max			Not used
Pos		0	Unchecked = 0, Checked = 1
Reply			40=ID~0 for Unchecked, 1 for Checked

RADIO

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**08**		**Radio**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		150	Numeric
Height		150	Numeric
Text			Each Buttons Text separated by “|”
Just		1	Text of Left = 1, Text on Right = 2
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size		14	Font Size
Style		0	Font Style - Normal = 0, Bold = 1, talic = 2, Underline = 4
Min		1	Horizontal = 1, Vertical = 2
Max		0	Indicates which button is ON
Pos		0	Off = 0, ON = 1
Reply			40=ID~1 (0 = First button,1 = Second Button) …..

SHAPE

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**09**		**Shape**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		150	Numeric
Height		150	Numeric
Text			Not Used
Just		1	Square = 1, Circle = 2
Back		236236236	Background Color (RGB)
Color		000000000	Drawing Color (RGB)
Size			Not used
Style			Not used
Min		1	Non-Transparent = 1, Transparent = 2
Max			Not used
Pos			Not used
Reply			40=ID~1

SLIDER

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**10**		**Slider**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		200	Numeric
Text			Not Used
Just		1	Horizontal = 1, Vertical = 2
Back		236236236	Background Color (RGB)
Color		000000000	Foreground Color (RGB)
Size			Not Used
Style			Not Used
Min		0	Minimum Value
Max		100	Maximum Value
Pos		50	Current Value
Reply			40=ID~Current Position of Slider

IMAGE

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**11**		**Image**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		500	Numeric
Text			File Name (if file is in Client) or URL
Just			Not used
Back			Not used
Color			Not used
Size			Not used
Style			Not used
Min			Not used
Max			Not used
Pos			Not used
Reply			40=ID~1 fort short press, 2 for long press

WEB

↑

PARAM	TYPE	DEFAULT	COMMENT
Type	**12**		**Web**
ID			AlphaNumeric - Must be unique
Left		0	Numeric
Top		0	Numeric
Width		500	Numeric
Height		200	Numeric
Text			URL
Just			Not Used
Back			Not Used
Color			Not Used
Size			Not Used
Style			Not Used
Min			Not Used
Max			Not Used
Pos			Not Used
Reply			40=ID~URL of the Web address navigated to

UTILITY COMMANDS:

↑

These Commands do NOT create widgets. They are provided for utility purposes

Delete

PARAM	TYPE	DEFAULT	COMMENT
Type	**50**		**Delete**
ID			'Delete'
Text			The ID of the Widget to be deleted from the Page
Reply			If OK = 40, If Widget does not exists = 96

Example: 50=Delete~Text=MyButton

Load

PARAM	TYPE	DEFAULT	COMMENT
Type	**51**		**Load**
ID			'Load'
Text			TheFileName of the Template stored on the Client
Just		0	0 = Clear, 1 = Overlay
Reply			If OK = 40, If file does not exists = 96

Example: 51=Load~Text=Template1.dat
NOTE: Using this will NOT update config.py

Write

PARAM	TYPE	DEFAULT	COMMENT
Type	**52**		**Write**
ID			'Write''
Text			The Filename that the Template will be written to
Reply			If OK = 40, If FileName already exists = 96

Example: 52=Write~Text=NewTemplate.dat

Exit

PARAM	TYPE	DEFAULT	COMMENT
Type			**Exit**

Exit the App

INTERNALS ↑

config.py - This module holds the following variables :

LOT	This is a List of Dictionaries, where each Dictionary holds a Widget, and each Key/Value pair contains the Widget's current properties.
W	The Screen width of the current Client
H	The Screen Height of the current Client
Port	The current Communication Port
IP	The IP of the Host
Keys	The 4 Arrow-Key keyboard values for the keyboard of the Server (Used by the Generator)

EasyGUI.py -

This module contains the various functions described under COMMANDS
This is an optional Module, and should you want to code the Server by yourself from scratch, you are free to do so.

Since this module is provided in Source format, you are welcome to modify it, after which any responsibility for the results will be yours

SETUP

PREREQUISITES (for python):

python 3.x
Tkinter (pip install Tkinter)
keyboard (pip install keyboard)
messagebox (pip install messagebox)

All the necessary files will be contained in EasyGUI.zip.

1 - Unzip EasyGUI.zip into a Folder created for EasyGUI

2 - The following files will be created:

Setup.py	- This script will create config.py
EasyGen.py	- The Generator
01.gif to 12.gif	- Help files
EasyGUI.py	- The Command module
SimpServer.py	- The simplest possible Server example

3 - Run Setup.py (python Setup.py)

In order to be able to use the arrow keys in EasyGen which seems to be different on some operating systems the keyboard module need to be installed: (pip install keyboard)
Setup.py will prompt you to press each of the arrow keys in turn, the results of which will be stored in config.py

4 - The first time you run EasyGUI (or any time you change the Servers IP) the following screen will be displayed, and you need to type in the last three digits of the Server's IP (left zero filled) and the Port number (5 digits, left zero filled).
Ie: If the Server's IP is 192.145.23.14 and the Port number is 5200, you should type in: 01405200