■■■■■■■■■■ - ■■■■

1. ■■■■

■■■■■■■■■■■■■■■■ Scratch ■■

1.1 ■■■■

- ■■■■■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■■■■■

1.2 ■■■■■

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                    ■■■■■                          ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  ■
■  ■                ■■ (index.html)                 ■  ■
■  ■  ■■■■■■■■■■■  ■■■■■■■■■■■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  ■  ■
■  ■  ■ ■■■  ■  ■ ■■■  ■  ■     ■■■■ & ■■      ■  ■  ■
■  ■  ■■■■■■■■■■■  ■■■■■■■■■■■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  ■  ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                        ■
                        ■ HTTP POST /api/control
                        ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                ■■■■■■■ (server.py)                ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■  ■  ThreadingHTTPServer                            ■    ■
■  ■  ■■■ ■■■■■■ (HTML/CSS/JS)                    ■    ■
■  ■  ■■■ /api/control ■■■■                        ■    ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                        ■
                        ■ HTTP POST /control
                        ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                ■■■■ (ESP32 + MicroPython)            ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■  ■  RobotWifi (robot_wifi.py)                       ■    ■
■  ■  ■■■ HTTP Server (■■ 80)                        ■    ■
■  ■  ■■■ ■■■■ & ■■                               ■    ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■                         ■                              ■
■                         ▼                              ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■  ■  Quad (quad.py)                                   ■    ■
■  ■  ■■■ 8 ■■■■■                                  ■    ■
■  ■  ■■■ ■■■■ (forward, backward, turn_L, dance...)     ■    ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■    ■
■                            ■                                  ■
■                            ▼                                  ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■    ■
■  ■  Oscillator (oscillator.py)                           ■    ■
■  ■  ■■■ PWM ■■■■■■■                                 ■    ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■    ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

2. ■■■■■■

2.1 ■■■

- ■■■■■■HTML5 + CSS3 + Vanilla JavaScript
- ■■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■■■■■ HTML ■■■

2.2 ■■■■

2.2.1 ■■■■■■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■

```
const BLOCKS = [
  { cat: 'motion', type: 'forward', title: '■■', subtitle: '■■■', className: 'c-motion', badge: '■■' },
  { cat: 'motion', type: 'backward', title: '■■', subtitle: '■■■', className: 'c-motion', badge: '■■' },
  { cat: 'motion', type: 'turn_L', title: '■■', subtitle: '■■■', className: 'c-motion', badge: '■■' },
  { cat: 'motion', type: 'turn_R', title: '■■', subtitle: '■■■', className: 'c-motion', badge: '■■' },
  { cat: 'control', type: 'wait', title: '■■', subtitle: '■■■■■', className: 'c-control', badge: '■■' },
  { cat: 'control', type: 'repeat', title: '■■', subtitle: '■■■■', className: 'c-control', badge: '■■' },
  // ... ■■■■■■
]
```

■■■■■

■■	■■	■■
motion	■■■■■	■■ (#4c97ff)
control	■■■■■	■■ (#ffab19)
events	■■■■■	■■ (#ffd500)

2.2.2 ■■■■ (Drag & Drop)

■■ HTML5 ■■■■ API ■■■

■■■■■

```
■■■■■■■■■■■■■■■■■    dragstart     ■■■■■■■■■■■■■■■■■    dragover     ■■■■■■■■■■■■■■■■■
■   ■■■     ■ ■■■■■■■■■■■■■■■■ ■   ■■■     ■ ■■■■■■■■■■■■■■■■ ■   ■■■     ■
■  (palette)   ■                 ■  (payload)   ■                 ■  (stack)     ■
■■■■■■■■■■■■■■■■■                ■■■■■■■■■■■■■■■■■                 ■■■■■■■■■■■■■■■■■
       ■                                                                  ■
       ■  ■■ dataTransfer                                                 ■
       ■  { kind: 'palette', type: 'forward' }                            ■
       ▼                                                                  ▼
  ■■■■■■■                                                    drop ■■■■
```

■■■■■■■

```
// ■■■■■■■
el.addEventListener('dragstart', (e) => {
  const payload = JSON.stringify({ kind: 'palette', type: blockDef.type })
  e.dataTransfer.setData('text/plain', payload)
  e.dataTransfer.effectAllowed = 'copy'
})

// ■■■■■■■
stackEl.addEventListener('drop', (e) => {
  const payload = tryReadPayload(e.dataTransfer.getData('text/plain'))
  if (payload.kind === 'palette') {
    const inst = createInstance(payload.type, {})
    stackEl.appendChild(inst)
  }
})
```

■■■■■

- `copy`■■■■■■■■■■■■■■■■■■■■
- `move`■■■■■■■■■■■■■

2.2.3 ■■■■■

■■■■■■■■■■■■■■■■■■■■■■

```
function compileStack(stackEl, ctx) {
  const children = Array.from(stackEl.querySelectorAll(':scope > .inst'))
  const out = []
  for (const el of children) {
    const type = el.dataset.type
    if (type === 'start') continue
    if (type === 'wait') {
      const ms = clampInt(input.value, 0, 60000, 0)
      out.push({ kind: 'wait', ms })
      continue
    }
    if (type === 'repeat') {
      const times = clampInt(input.value, 1, 20, 1)
      const inner = compileStack(childStack, ctx)
      for (let i = 0; i < times; i++) {
        out.push(...inner)
      }
      continue
    }
    out.push({ kind: 'command', command: type })
  }
  return out
}
```

■■■■■■■

```
// ■■■■■■■ → ■■(500ms) → ■■
[
  { kind: 'command', command: 'forward' },
  { kind: 'wait', ms: 500 },
  { kind: 'command', command: 'turn_L' }
]
```

2.2.4 ■■■■■

■■■■■■■

```
let running = false
let currentRunState = null

// ■■■■■■
const runState = {
  stop: false,           // ■■■■
  timers: [],            // ■■■■■■■■■■■
  abortControllers: []   // AbortController ■■■■■■■■■■
}
```

■■■■■

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                        runProgram()                              ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■  1. compileProgram() → ■■■■■■■■■                            ■
■  2. ■■ running = true, ■■■■■■                              ■
■  3. for (step of steps):                                         ■
■     ■■■ step.kind === 'wait' → sleep(step.ms)                    ■
■     ■■■ step.kind === 'command' → postCommand(step.command)      ■
■  4. ■■■■■■■ UI                                              ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

■■■■■■■

```
async function postCommand(command, runState) {
  const s = readSettings()
  let url, body

  if (s.useProxy) {
    // ■■■■■■■■■■
    url = '/api/control'
    body = JSON.stringify({ baseUrl: s.baseUrl, command })
  } else {
    // ■■■■■■■■■■■■ CORS■
    url = (s.baseUrl || '') + '/control'
    body = JSON.stringify({ command })
  }

  // ■■■■■■■■■■
  const res = await fetchWithTimeout(url, opts, s.timeoutMs, runState)
  return res
}
```

2.2.5 ■■■■■

■■ localStorage ■■■■■■■

```
const LS_KEYS = {
  baseUrl: 'quad_robot_base_url',    // ■■■■■
  timeout: 'quad_http_timeout_ms',   // ■■■■
  retry: 'quad_http_retry',          // ■■■■
  useProxy: 'quad_use_proxy'         // ■■■■■■
}
```

3. ■■■■■■

3.1 ■■■■■■■ (server.py)

3.1.1 ■■■■

```
class Handler(SimpleHTTPRequestHandler):
    def do_POST(self):
        if self.path.rstrip('/') != '/api/control':
            return  # 404

        # 1. ■■■■
        req = json.loads(raw)
        command = req.get('command')
        base_url = req.get('baseUrl')

        # 2. ■■■■■■
        target = f'{base_url}/control'
        body = json.dumps({'command': command})

        # 3. ■■■■■■■■■
        with urllib.request.urlopen(r, timeout=15) as resp:
            # ■■■■■■■
```

3.1.2 ■■■■■■■

```
■■■■■■■■ http://localhost:8001■■■■■■ http://192.168.2.182

■ ■■■■■■■■■■■■
■■■ → http://192.168.2.182/control
      ↑ CORS ■■■■■■
```

```
■ ■■■■■■■■■■■■
■■■ → http://localhost:8001/api/control■■■■■■■■
■■■■ → http://192.168.2.182/control■■■■■■■■ CORS ■■■
```

3.1.3 CORS ■■

```
def end_headers(self):
    self.send_header('Access-Control-Allow-Origin', '*')
    self.send_header('Access-Control-Allow-Methods', 'POST, OPTIONS')
    self.send_header('Access-Control-Allow-Headers', 'Content-Type')
    super().end_headers()

def do_OPTIONS(self):
    self.send_response(HTTPStatus.NO_CONTENT)
    self.end_headers()
```

3.2 ■■■■■■ (robot_wifi.py)

3.2.1 ■■■■

AP ■■■■■■■■■

```
ESP32 ■■■■ ←■■→ ■■/■■■■
IP: 192.168.2.182■■■■
```

STA ■■■■■■■■■

```
■■■ ←■■→ ESP32
        ↑
    ■■/■■■■■■■■■
```

3.2.2 HTTP ■■■

```
class RobotWifi:
    def create_server(self):
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_socket.bind(('', 80))
        server_socket.listen(128)

        while True:
            client_socket, addr = server_socket.accept()
            self.handle_request(client_socket)
```

3.2.3 ■■■■

```
def handle_request(self, client_socket):
    request = client_socket.recv(1024)
    method, path, _ = request_lines[0].split()

    if method == "POST" and path == "/control":
        # ■■■■■■■
        command = json.loads(post_data).get("command")
        method = getattr(self.robot, command)  # ■■■■■■■■■
```

```
            method()
            return json.dumps({"status": "200", "msg": command})
        else:
            # ■■■■■■
            return self.html
```

3.3 ■■■■■■■ (quad.py)

3.3.1 ■■■■

```
    ■■■■■■■■■■8 ■■■■■

        ■ (head)
        ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
        ■  FRH(■■■)    ■    FLH(■■■)       ■
        ■  Pin12         ■    Pin16            ■
        ■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■      ■
        ■  FRL(■■■)    ■    FLL(■■■)       ■
        ■  Pin25         ■    Pin18            ■
        ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
        ■  BRH(■■■)    ■    BLH(■■■)       ■
        ■  Pin13         ■    Pin17            ■
        ■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■      ■
        ■  BRL(■■■)    ■    BLL(■■■)       ■
        ■  Pin26         ■    Pin19            ■
        ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
        ■ (tail)
```

3.3.2 ■■■■■■

■■■■■■

■■■■■■■■■■■■■■■■■

```
    def oscillateServos(self, amplitude, offset, period, phase, cycle=1.0):
        for i in range(self._servo_totals):
            self._servo[i].SetO(offset[i])      # ■■■
            self._servo[i].SetA(amplitude[i])   # ■■
            self._servo[i].SetT(period[i])      # ■■
            self._servo[i].SetPh(phase[i])      # ■■

        # ■■■■■■■■■
        while x <= period[0] * cycle + ref:
            for i in range(self._servo_totals):
                self._servo[i].refresh()
```

■■■■■■■

■■	■■	■■
amplitude	■■■■■■■■	[15, 15, 20, 20, ...]
offset	■■■■■■■■■■■	[0, 0, -15, 15, ...]

period	■■■■■■■■■■■■■	[800, 800, 400, 400, ...]
phase	■■■■■■■■■■■■■	[0, 0, 90, 90, ...]

3.3.3 ■■■■■■

■■■■ (forward)■

```
def forward(self, steps=3, t=800):
    x_amp = 15      # X ■■■■
    z_amp = 15      # Z ■■■■■■■■■■
    ap = 10         # ■■■■
    hi = 15         # ■■■■

    amplitude = [x_amp, x_amp, z_amp, z_amp, x_amp, x_amp, z_amp, z_amp]
    offset = [
        0 + ap - front_x,   # FRH
        0 - ap + front_x,   # FLH
        0 - hi,             # FRL
        0 + hi,             # FLL
        0 - ap - front_x,   # BRH
        0 + ap + front_x,   # BLH
        0 + hi,             # BRL
        0 - hi              # BLL
    ]
    phase = [0, 0, 90, 90, 180, 180, 90, 90]

    self._execute(amplitude, offset, period, phase, steps)
```

■■■■■

```
■■■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

FRH: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 0°)
FLH: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 0°)
FRL: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 90°)
FLL: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 90°)
BRH: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 180°)
BLH: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 180°)
BRL: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 90°)
BLL: ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  (■■ 90°)
```

4. ■■■■

4.1 API ■■

4.1.1 ■■■■

■■■

```
POST /control HTTP/1.1
Content-Type: application/json

{
  "command": "forward"
}
```

■■■

```
{
  "status": "200",
  "msg": "forward"
}
```

4.1.2 ■■■■

■■■

```
POST /api/control HTTP/1.1
Content-Type: application/json

{
  "baseUrl": "http://192.168.2.182",
  "command": "forward"
}
```

■■■■■■■■■■■■■

4.2 ■■■■■

■■	■■	■■
`forward`	■■	steps=3, t=800
`backward`	■■	steps=3, t=800
`turn_L`	■■	steps=2, t=1000
`turn_R`	■■	steps=2, t=1000
`omni_walk`	■■	steps=2, t=1000
`home`	■■■■	-
`dance`	■■	steps=3, t=2000
`hello`	■■■	-
`up_down`	■■■	steps=2, t=2000
`push_up`	■■■	steps=2, t=2000

`front_back`	■■■	steps=2, t=1000
`wave_hand`	■■■	steps=3, t=2000
`scared`	■■	-
`moonwalk_L`	■■■	steps=4, t=2000

5. ■■■■

5.1 ■■■■

```
#!/bin/zsh
# start-workbench.sh

robot_url="${1:-}"      # ■■■■■■■■■■■
port="${2:-8001}"       # ■■■■■■■■■
host="0.0.0.0"          # ■■■■

# ■■■■■■
if [[ -n "$robot_url" ]]; then
  export ROBOT_BASE_URL="$robot_url"
fi

# ■■■■
python3 workbench/server.py --port "$port" --host "$host"
```

5.2 ■■■■

■■■■■■■■■■

```
./start-workbench.sh http://192.168.2.182 8001
```

■■■■■■■■

```
python3 workbench/server.py --port 8001 --host 0.0.0.0
```

■■■■■■■■■

```
ROBOT_BASE_URL=http://192.168.2.182 python3 workbench/server.py --port 8001
```

5.3 ■■■■

```
http://127.0.0.1:8001/?robot=http://192.168.2.182
```

URL ■■ `robot=` ■■■■■■■■■■■ Base URL■

6. ■■■■

6.1 ■■■■■■

```
■■■■■■■■■■■   ■■■■    ■■■■■■■■■■■   ■■    ■■■■■■■■■■■
■  ■■   ■ ■■■■■■■■■■■■■■ ■  ■■■  ■ ■■■■■■■■■ ■ ■■■■ ■
■■■■■■■■■■■             ■■■■■■■■■■■         ■■■■■■■■■■■
                                            ■
                                            ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                   ■■■■                          ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■  ■  for step of steps:                         ■   ■
■  ■    ■■■ wait → sleep(ms)                     ■   ■
■  ■    ■■■ command → POST /api/control          ■   ■
■  ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                              ■
                              ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■  POST /api/control                                        ■
■  Body: { "baseUrl": "http://192.168.2.182", "command": "forward" }■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                              ■
                              ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■  server.py (■■)                                          ■
■  1. ■■ baseUrl ■ command                                 ■
■  2. POST http://192.168.2.182/control                     ■
■  3. ■■■■                                                 ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                              ■
                              ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■  robot_wifi.py (■■■■)                                    ■
■  1. ■■ command                                            ■
■  2. getattr(robot, command)()                             ■
■  3. ■■ {"status": "200", "msg": "forward"}               ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                              ■
                              ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■  quad.py (■■■■)                                          ■
■  1. ■■■■■■■                                            ■
■  2. ■■■■■■                                             ■
■  3. ■■■■                                                 ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

7. ■■■■

```
quad-mpy/
```

```
■■■ workbench/
■   ■■■ index.html         # ■■■■■■■■■■■■■
■   ■■■ index-apple.html   # ■■■■■■
■   ■■■ server.py          # ■■■■■■■
■   ■■■ README.md          # ■■■■
■■■ quad.py               # ■■■■■■■■
■■■ robot.py              # ■■■■■
■■■ robot_wifi.py         # WiFi ■■■ HTTP ■■
■■■ oscillator.py         # ■■■■■
■■■ main.py               # ■■■■■
■■■ index.html            # ■■■■■■■■
■■■ start-workbench.sh    # ■■■■
```

8. ■■■■

8.1 ■■

1. ■■■■■■■■■■■■■■■■
2. ■■■■■■■■■■■■■■■
3. ■■■ **CSS**■■■ CSS ■■■■■■
4. ■■■■■■■■■■■■■■■■■■■■

8.2 ■■

1. ■■■■■■ 130 ■■■■■■■■■■■
2. ■■■■■■■ ThreadingHTTPServer ■■■■
3. ■■■■■■■■ CORS ■■
4. ■■■■■■■■■■■■■■■■■

8.3 ■■■■

1. ■■■■■■■ `getattr` ■■■■■■■■
2. ■■■■■■■■■■■■■■■
3. ■■■■■■■■ AP ■ STA ■■■■■■

9. ■■■■

9.1 ■■■■■

■ quad.py ■■■■■■■

```
def new_action(self, steps=2, t=1000):
    amplitude = [...]
    offset = [...]
    phase = [...]
    self._execute(amplitude, offset, period, phase, steps)
```

■■■ BLOCKS ■■■■■■■■■■

```
{ cat: 'motion', type: 'new_action', title: '■■■', subtitle: '■■', className: 'c-motion', badge: '■■' }
```

9.2 ■■■■■■■

1. ■ BLOCKS ■■■■■■
2. ■ createInstance() ■■■■■ UI■■■■■■■
3. ■ compileStack() ■■■■■■■

10. ■■■■

■■	■■	■■■■
1.0	-	■■■■■■■■■■■■■
1.1	-	■■■■■■■■■■■■■
1.2	-	■■■■■■■■■■■■■