F.O.C.U.S.

MOMENTUM

Physical Task Management with LED Progress Visualization

Version 1.0.1	Raspberry Pi 4	MIT License

Documentation Manual

Find the complete source code and project files on GitHub:

github.com/stoptexting/couad-focus

Table of Contents

1. Introduction

F.O.C.U.S. MOMENTUM is a unique task management system that bridges the gap between digital project management and physical, tangible feedback. Built on the Raspberry Pi 4 platform, it combines a modern web interface with real-time progress visualization on a 64x64 RGB LED matrix display.

The system provides hierarchical task management following the structure of Projects, Sprints, User Stories, and Tasks, with automatic LED updates as work progresses. This creates a physical manifestation of productivity that can serve as both motivation and accountability.

Why Physical Visualization?

Digital dashboards are powerful, but they require active engagement. A physical LED display provides passive, ambient awareness of project status. Team members can glance at the display from across the room to instantly understand progress without interrupting their workflow.

2. System Overview

The F.O.C.U.S. MOMENTUM system consists of several integrated components working together to provide a seamless experience from web interface to physical display.

Core Components

Component	Description
Web Frontend	Next.js 16 application with React 19, featuring a neobrutalist design with sharp corners, bold typography, and hard shadows.
Backend API	Flask 3.0 REST API with SQLAlchemy ORM, handling all business logic and database operations.
LED Manager	Python daemon with priority queue and exclusive hardware access via Unix socket IPC.
Boot Manager	System orchestrator handling WiFi connection, Ngrok tunnels, Discord bot, and service management.
Discord Bot	Remote control interface for system management, status monitoring, and command execution.

Data Flow

The system follows a clear data flow pattern from user action to physical display:

1. User performs an action in the web frontend (creates task, updates status, etc.)
2. Frontend sends API call to Flask backend
3. Backend updates SQLite database via SQLAlchemy
4. Progress percentages are automatically recalculated
5. LED update command sent via Unix socket to LED daemon
6. RGB matrix displays updated progress visualization

3. Features & Capabilities

Feature	Description
Hierarchical Tasks	Project → Sprint → User Story → Task structure for organized management
LED Progress	4 display layouts: single, sprint view, user story layout, and cycling mode
Real-time Sync	Automatic LED updates immediately when tasks are completed
Taiga Integration	Webhook-based synchronization with Taiga project management tool
Discord Bot	Remote control via Discord commands for system management
Auto-boot	Automatic service startup when Raspberry Pi powers on
Tunnel Access	Ngrok SSH and HTTP tunnels for remote access from anywhere
Neobrutalist UI	Modern design with sharp corners, bold typography, and hard shadows

4. System Architecture

The F.O.C.U.S. MOMENTUM architecture is designed for reliability, modularity, and ease of maintenance. Each component operates independently but communicates through well-defined interfaces.

External Access Layer

The system can be accessed remotely through multiple channels. Taiga Server provides project management integration via webhooks. Discord API enables remote command and control. Ngrok tunnels provide secure SSH and HTTP access from any location.

Boot Manager Layer

The Boot Manager is the system orchestrator that runs at startup. It follows an 11-step boot sequence:

1. WiFi connection establishment and verification
2. Ngrok tunnel initialization for SSH access
3. Ngrok tunnel initialization for HTTP access
4. Discord bot connection and authentication
5. LED Manager daemon verification
6. Backend service startup
7. Frontend service startup
8. Nginx proxy configuration
9. Health check verification
10. Status notification to Discord
11. System ready confirmation

Web Layer

Nginx acts as a reverse proxy on port 80, routing requests to the appropriate backend. The root path serves the Next.js frontend on port 3000, while API paths are forwarded to the Flask backend on port 5001.

LED Manager Daemon

The LED Manager runs as a dedicated daemon with exclusive access to the LED hardware. It implements a priority queue system (HIGH, MEDIUM, LOW) to ensure critical updates are displayed immediately. Communication uses a JSON protocol over Unix sockets for fast, reliable IPC.

5. Hardware Requirements

Required Components

Component	Specification	Purpose
Raspberry Pi 4	2GB+ RAM	Main processor
MicroSD Card	16GB+ Class 10	System storage
Power Supply	5V 3A USB-C	Pi power
LED Matrix	64x64 RGB HUB75E	Display
Matrix Power	5V 8A (minimum)	LED power
Jumper Wires	Female-Female	GPIO connections

LED Matrix Specifications (HUB75E)

- Resolution: 64x64 pixels (4,096 LEDs total)
- Pitch: 3mm
- Physical Size: 192 x 192 x 14mm
- Interface: HUB75E (1/32 scan)
- Operating Voltage: 5V DC
- Maximum Power: 40W

Power Configuration

The system requires two separate power supplies. The Raspberry Pi 4 connects via USB-C to a 5V 3A power supply. The LED Matrix requires a dedicated 5V 8A+ power supply connected via barrel or screw terminals.

IMPORTANT: You must connect the ground (GND) from both power supplies together to ensure proper operation.

6. LED Display Layouts

The LED matrix supports four distinct display layouts, each designed for different viewing contexts and information needs.

Single View

The Single View displays the project name prominently with a horizontal progress gauge showing overall completion percentage. This layout is ideal for quick status checks and provides at-a-glance understanding of project health. The display includes the current sprint number and completed user stories count.

Sprint View

Sprint View shows the overall project progress as a horizontal bar at the top, with three vertical sprint progress bars below. Each sprint bar fills from bottom to top as tasks are completed, providing visual comparison between sprint velocities. This layout is perfect for sprint planning meetings and team standups.

User Story Layout

This layout focuses on the current sprint, showing sprint progress at the top with two horizontal user story progress bars below. It allows teams to see which user stories are progressing and which may need attention. The compact format fits more detail about the active work.

Cycling Mode

Cycling Mode automatically rotates through user stories in pairs at configurable intervals. This provides comprehensive visibility into all active work without requiring manual navigation. It is ideal for display in common areas where team members pass by regularly.

7. Installation Guide

Prerequisites

- Raspberry Pi 4 (2GB+ RAM)
- 64x64 RGB LED Matrix (HUB75E interface)
- Raspberry Pi OS Lite (Debian 12 Bookworm)
- Python 3.11+
- Node.js 18+

Quick Start (4 Steps)

Step 1: Clone Repository

Clone the F.O.C.U.S. repository from GitHub to your Raspberry Pi.

Step 2: Run Installation Script

Execute the installation script with sudo privileges. This will install all system dependencies, set up virtual environments, build the frontend, and configure systemd services.

Step 3: Configure Credentials

Create the configuration directory and copy the secrets template. Fill in your Discord bot token, Ngrok authentication, and WiFi credentials.

Step 4: Reboot

Reboot the Raspberry Pi to start all services automatically. The boot manager will handle WiFi connection, tunnel setup, and service initialization.

Enabling SPI Interface

The LED matrix requires SPI to be enabled on the Raspberry Pi. Use raspi-config to enable the SPI interface, then verify by checking that /dev/spidev0.0 and /dev/spidev0.1 exist.

8. Configuration

Configuration Files

File	Location	Purpose
`secrets.env`	`~/.config/bootmanager/`	Discord, Ngrok, WiFi credentials
`config.json`	`server/`	LED zone configuration
`.env`	`server/`	Flask environment
`.env.local`	`frontend/`	Frontend API URL

secrets.env Configuration

The secrets.env file contains all sensitive credentials and is required for the system to function. You must configure the following sections:

Discord Bot Settings

Set your Discord bot token (DISCORD_BOT_TOKEN) and the channel ID where the bot should respond (DISCORD_CHANNEL_ID).

Ngrok Settings

Configure your Ngrok authentication token (NGROK_AUTH_TOKEN) and optional HTTP basic authentication credentials (NGROK_HTTP_USERNAME, NGROK_HTTP_PASSWORD).

WiFi Settings

Set the network name (WIFI_SSID) and password (WIFI_PASSWORD) for the WiFi network the system should connect to.

9. Discord Bot Commands

The Discord bot provides remote control and monitoring capabilities for the F.O.C.U.S. system. All commands are prefixed with an exclamation mark (!).

Command Reference

Command	Description
`!<cmd>`	Execute any shell command on the Raspberry Pi
`!ps`	List all active background jobs
`!tail <id> [n]`	Show last n lines of a job's output (default: 20)
`!stop <id>`	Stop a running background job
`!status`	Display system status (CPU, RAM, uptime)
`!services`	Show status of all F.O.C.U.S. services
`!urls`	Display SSH and HTTP tunnel URLs
`!restart-server`	Restart the Flask backend service
`!restart-frontend`	Restart the Next.js frontend service
`!reboot`	Reboot the Raspberry Pi
`!help`	Display help message with all available commands

Setting Up the Discord Bot

- Go to the Discord Developer Portal and create a new application
- Navigate to Bot section and add a bot to your application
- Enable "Message Content Intent" under Privileged Gateway Intents
- Copy the bot token and add it to your secrets.env file
- Use OAuth2 URL Generator with "bot" scope and required permissions
- Invite the bot to your server using the generated URL
- Copy your channel ID and add it to secrets.env

10. Taiga Integration

F.O.C.U.S. MOMENTUM integrates with Taiga, the open-source project management tool, to synchronize task data and receive real-time updates via webhooks.

How It Works

When you update a task in Taiga (change status, complete a user story, etc.), Taiga sends a webhook notification to F.O.C.U.S. The system processes this webhook, updates the local database, recalculates progress percentages, and immediately updates the LED display to reflect the change.

Setting Up Webhooks in Taiga

To configure Taiga webhooks:

- Open your Taiga project and go to Settings
- Navigate to Integrations, then Webhooks
- Add a new webhook with your Ngrok URL followed by /api/taiga/webhook
- Generate a secure secret and save it
- Enter this same secret in the F.O.C.U.S. Settings page

Auto-Login Feature

When enabled, the system will automatically authenticate with Taiga on boot using stored credentials. This allows the system to sync data and display progress without any manual intervention after power cycles.

11. Troubleshooting

LED Not Displaying

- Verify SPI is enabled by checking for /dev/spidev devices
- Confirm user is in gpio and video groups
- Check LED Manager daemon is running with systemctl status
- Verify the Unix socket exists at /tmp/led-manager.sock
- Test LED connection manually using the LED client library

WiFi Connection Issues

- Verify credentials in secrets.env are correct
- Check NetworkManager status with nmcli device status
- Test manual connection using nmcli device wifi connect

Discord Bot Offline

- Confirm bot token is correct and not expired
- Ensure Message Content Intent is enabled in Discord Developer Portal
- Verify channel ID is correct
- Check bootmanager logs for Discord-related errors

Service Won't Start

- Check service status with systemctl status <service-name>
- View detailed logs with journalctl -u <service-name>
- Reset failed state with systemctl reset-failed

Log Locations

Component	Log Location
LED Manager	`/var/log/led-manager.log`
Boot Manager	`/var/log/bootmanager.log`
Flask Server	`server/bootmanager_server.log`
Job Outputs	`~/.cmdruns/<job_id>.log`

12. Source Code & License

GitHub Repository

The complete source code, hardware schematics, and project documentation are available on GitHub. You can clone the repository, submit issues, or contribute improvements.

github.com/stoptexting/couad-focus

Technology Credits

- rpi-rgb-led-matrix by Henner Zeller - LED matrix control library
- shadcn/ui - React component library
- TanStack Query - Data fetching and caching
- discord.py - Discord bot framework
- Taiga - Open source project management

MIT License

F.O.C.U.S. MOMENTUM

Physical Task Management with LED Progress Visualization