🧠 CLAWBOLT

A Telegram-Controlled Antigravity AI Agent with Real-Time Screen Awareness

“Vibe coding meets remote AI control.”

1. PROJECT OVERVIEW

CLAWBOLT is a Telegram-driven control layer that:

- Forwards user messages into Antigravity AI’s “Ask anything” input box
- Tracks Antigravity’s dynamic UI movement
- Streams real-time AI activity (screenshots, progress, errors)
- Executes system & agent commands independently of AI quota limits
- Handles password prompts (screen lock & keyring) remotely via Telegram
- Survives Antigravity restarts, crashes, and system reboots

2. CORE CONCEPT

Message Flow (Normal Chat)

Telegram User
↓
Telegram Bot
↓
CLAWBOLT Router
↓
Context Injector
↓
Antigravity “Ask anything” Input Box
↓

Antigravity Processing
 ↓
Live Screen Capture + Status
 ↓
send_telegram.py (Realtime Updates)

Command Flow (

/command

)

Telegram User
 ↓
Command Parser
 ↓
Agent Script (Independent of AI)
 ↓
System / Screen / Files / State
 ↓
send_telegram.py

---

3. DEFAULT CONTEXT INJECTION

Before every normal message, CLAWBOLT injects:

Context:
Your response will be sent via send_telegram.py.
Respond with clear steps.
If processing takes time, output progress.
If errors occur, explain the cause.

User Message Format

What programming language? ␣⏎
 Space + Enter triggers send → avoids premature injection

---

4. ANTIGRAVITY INPUT BOX PROBLEM (CRITICAL)

Problem

- “Ask anything” input box:
    - Moves location dynamically
    - Changes window hierarchy
    - Fresh launch ≠ continuous session

CLAWBOLT SOLUTION (Hybrid Detection)

1.

Visual Anchoring

- OCR scan for:
    - "Ask anything"
    - "Type your question"
- Bounding box detection

2.

UI Tree Scanning

- Linux tools:
    - xdotool
    - wmctrl
    - xprop
    - pyatspi

3.

Fallback Mode

- If input box not found:

    - Screenshot sent to Telegram
    - Status: WAITING_FOR_INPUT_BOX
    - Retry every X seconds

4.

Manual Override

- /screen
- User visually confirms state

---

5. REAL-TIME AI PROCESS STREAMING

Every Antigravity response produces:

- 📸 Screenshot
- 🧠 Status message
- 🔄 Progress update
- ❌ Error detection (if any)

Update Frequency

- Adaptive:
    - Idle → every 5s
    - Typing → every 1s

    - Error → immediate

---

6. TELEGRAM COMMAND SYSTEM

Commands do not rely on Antigravity AI

/rules

Displays all agent commands.

/report

System report:

- OS version
- CPU / RAM
- Disk
- GPU
- Antigravity status
- CLAWBOLT uptime

/screen

- Instant screenshot
- Auto-sent to Telegram

/hear <value>

Sets:

- Monitoring sensitivity
- Screen polling rate
- Input detection threshold

/watch <time>

Screen recording:

/watch 30s
/watch 2m

- Saves video
- Sends to Telegram

/restart

- Closes Antigravity
- CLAWBOLT remains alive
- Polls until Antigravity reopens
- Rebinds input mapping

/sysrest

System reboot:

- Requires confirmation:

YES

NO

- On reboot:
    - CLAWBOLT auto-starts
    - Antigravity auto-launches

/syslogout

Logs out current user session

Requires YES/NO validation

/ls <path>

Remote file listing.

/save

- Zips current CLAWBOLT state
- Logs
- Config
- Screens
- Sends zip to Telegram

/quota

- Displays:

    - AI model
    - Rate limits
    - Token estimates
- Uses local quota detection lib

---

7. PASSWORD & KEYRING DETECTION (SECURITY CORE)

Triggers

- Screen lock wake
- Antigravity keyring popup
- System authentication dialog

Detection Method

- OCR keywords:
    - “Password”
    - “Authentication required”
    - “Keyring”
- Window class detection

Flow

Password Dialog Detected
 ↓
Telegram Notification
 ↓
User Sends Password
 ↓

Secure Injection
↓
Continue Execution

⚠️ Passwords:

- Never logged
- Memory-only
- Immediately wiped after use

---

8. TELEGRAM AUTHENTICATION

Required:

- Bot Token
- Allowed User ID
- Optional: Admin list

Unauthorized users:

- Ignored
- Logged
- Optional alert

---

9. TECHNOLOGY STACK

Language

- Python 3.11+

Core Libraries

- python-telegram-bot
- pyautogui
- opencv-python
- pytesseract
- Pillow
- xdotool
- wmctrl
- psutil
- watchdog
- ffmpeg

---

10. DIRECTORY BLUEPRINT

```
CLAWBOLT/
│
├── core/
│   ├── router.py            # Message & command dispatcher
│   ├── context_injector.py
│   ├── antigravity_mapper.py  # Dynamic input box locator
│   ├── screen_watcher.py
│   ├── password_detector.py
│
├── agent/
│   ├── rules.py
│   ├── report.py
│   ├── screen.py
│   ├── watch.py
│   ├── restart.py
│   ├── sysrest.py
│   ├── syslogout.py
│
```

```
│   ├── ls.py
│   ├── save.py
│   └── quota.py
│
├── telegram/
│   ├── bot.py
│   ├── send_telegram.py
│   └── auth.py
│
├── system/
│   ├── autostart.service
│   ├── antigravity_launcher.sh
│   └── recovery.py
│
├── storage/
│   ├── screenshots/
│   ├── recordings/
│   ├── logs/
│   └── states/
│
├── config/
│   ├── settings.yaml
│   └── secrets.env
│
├── docs/
│   ├── architecture.md
│   ├── commands.md
│   ├── security.md
│   └── roadmap.md
│
└── main.py
```

---

11. ROADMAP (PHASED)

PHASE 1 – Foundation

- Telegram bot
- Auth
- /screen, /report

PHASE 2 – Antigravity Integration

- Input box mapping
- Context injection
- Screenshot streaming

PHASE 3 – Agent Commands

- Full /command set
- Independent execution

PHASE 4 – Security & Recovery

- Password detection
- Restart polling
- Crash recovery

PHASE 5 – UX & Optimization

- Adaptive screen rates
- Error visualization
- Logs & telemetry

---

12. WHAT MAKES CLAWBOLT DIFFERENT

✅ AI-independent control

✅ Real-time UI awareness

✅ Remote password handling

✅ Crash-proof agent

✅ Telegram as a full control console