INSTITUTE FOR DEFENSE ANALYSES

Making an Office Nerf Turret: A Vulnerable, Cyber-Kinetic Demonstration

Peter Mancini, Project Leader

Jason R. Schlup
Mark R. Herrera

April 2020
Approved for Public Release.
IDA Document NS D-13158
H 2020-000135

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a nonprofit corporation that operates three Federally Funded Research and Development Centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical, and analytic expertise.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-19-D-0001, Task C9096, Cyber Experimentation and Training Lab for the AE / CRP / Central Research Project. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
The IDA Technical Review Committee was chaired by Mr. Robert R. Soule and consisted of Christine Lewis; Daniel Porter; Dean Thomas; and Tye Botting from the Operational Evaluation Division

For more information:
Peter Mancini, Project Leader
pmancini@ida.org703-845-2496

Robert R. Soule, Director, Operational Evaluation Division
rsoule@ida.org • (703) 845-2482

Rigorous Analysis | Trusted Expertise | Service to the Nation

INSTITUTE FOR DEFENSE ANALYSES

IDA Document D-13158 NS

Making an Office Nerf Turret: A Vulnerable, Cyber-Kinetic Demonstration

The authors of this paper conducting developmental testing of the Office Nerf Turret

Peter Mancini, Project Leader

Jason R. Schlup
Mark R. Herrera

Executive Summary

Our Cyber Lab team within the Operational Evaluation Division (OED) developed a mini-tutorial on basic cybersecurity concepts for the 2020 Defense and Aerospace Test and Analysis Workshop (DATAWorks) conference. To make the results of cyber compromise more tangible, we designed and constructed a desktop Office Nerf Turret (ONT) to be the centerpiece of the tutorial.

This write-up is a supplement to the mini-tutorial, and explains how we built the ONT and some of the vulnerabilities incorporated into the demonstration. This document lists the hardware used to build the turret, including the 3D-printed firing mechanism. We also highlight useful tutorials that contributed to the development of the machine learning and tracking components of the ONT.

INTRODUCTION

Our Cyber Lab team within the Operational Evaluation Division (OED) developed a mini-tutorial on basic cybersecurity concepts for the 2020 Defense and Aerospace Test and Analysis Workshop (DATAWorks) conference.[1] The Cyber Lab's goal is to provide an overview of important cybersecurity concepts, such as phishing, open source intelligence gathering, network enumeration, and exploitation. While these topics are interesting and important, our team has noticed a tendency for people to divorce dire cyber consequences from real-world effects. To make the results of cyber compromise more tangible, we designed and constructed a desktop Office Nerf Turret (ONT) to be the centerpiece of the tutorial.

Conceptually, the ONT is an autonomous Nerf blaster mounted on a turret, and is notionally intended to keep unauthorized personnel out of its operating area. Using a web camera to observe its environment, the ONT performs facial recognition via machine learning, locates faces in its field of view, compares tracked faces to those in its face database, targets identified threats, and fires Nerf darts at them. In our mini-tutorial, an employee uploads a set of images for each person he allows to enter his office. The ONT recognizes these faces as friendly and does not activate or shoot at these people. If a person not in the database enters the office, the ONT detects, tracks, and engages the target.

For our mini-tutorial, the turret also serves a second purpose: to serve as a kinetic demonstration of how a cyber compromise can be used to create real-world effects. The ONT has been purposely designed to include several vulnerabilities that an adversary might use to compromise the turret. The tutorial demonstrates how an adversary might work through a network to ultimately compromise the ONT control computer and cause the turret to turn on its user.[2]

While we encourage you to review the mini-tutorial, the focus of this write-up is to explain how we built the ONT and some of the vulnerabilities incorporated into the demonstration. This document lists the hardware used to build the turret, including the 3D-printed firing mechanism. We also highlight useful tutorials that contributed to the development of the machine learning and tracking components of the ONT.

HARDWARE ASSEMBLY

The ONT is composed of three main components: the pan-tilt base which orients the turret towards its intended target, the firing mechanism that launches darts at targets, and the control group that detects and identifies targets and sends commands to both the

---

[1] Mancini, Peter M., Allison, Stacey L., Herrera, Mark R., Schlup, Jason R., and Tran, Kelly. 2020. Taking Down a Turret: Introduction to Cyber Operational Test and Evaluation. IDA Document NS D-10566.

[2] This is an example of an integrity takeover attack.

pan-tilt base and the firing mechanism. Figure 1 shows an image of the fully assembled turret hardware, with each of the main components highlighted; Table 1 lists the parts used to construct each component. The following sections detail each portion of the turret.

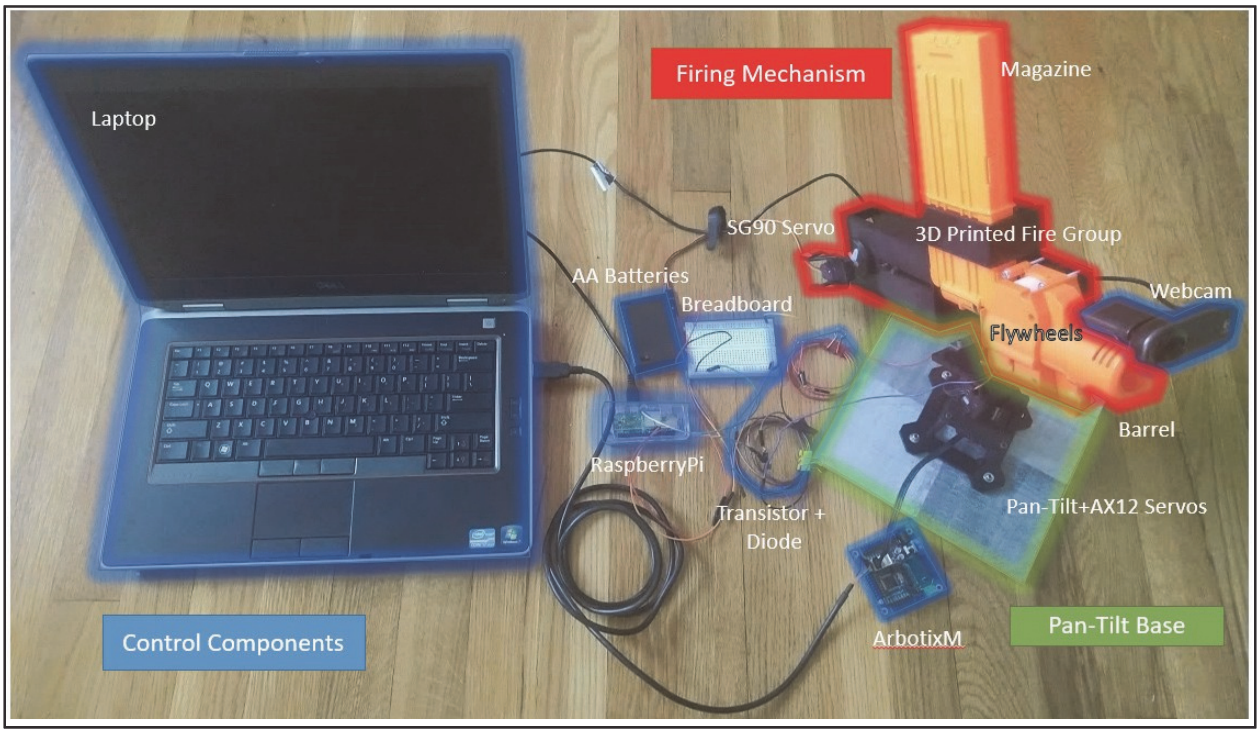

Figure 1. The fully assembled ONT with the pan-tilt base (green), firing mechanism (red), and control components (blue)

Table 1. List of hardware used to construct the ONT

TURRET COMPONENT	HARDWARE
Pan-Tilt Base	PhantomX Pan Tilt with two AX-12 ArbotiX-M servomotors
	Flooring tiles to anchor the base
Firing Mechanism	Nerf N-Strike Barricade RV-10
	3D printed turret components
	FUCAS 12-dart magazine
	3D printed slider crank mechanism
	SG90 servomotor
Control Group	Three AA batteries and housing
	1N4000 diode
	2N222A transistor
	Jumper cables
	320 ohm resistor
	ArbotiX-M robocontroller
	12-volt power supply
	Raspberry Pi Zero
	Breadboard
	Linux control laptop
	Logitech HD 720p webcam
	Micro Universal Serial Bus (USB) cable
	Universal asynchronous receiver-transmitter (UART) 6-pin to USB cable

Pan-Tilt Base

A PhantomX Pan Tilt, powered by two AX-12 ArbotiX-M servomotors, drives the turret motion (see Figure 2). These servomotors are well suited for this project; they can apply the necessary torque and range of motion to support the turret, and they use an easy-to-implement serial connection for command and control. In order to keep the turret from tipping on its side due to the motion of the pan-tilt base, we attached four flooring tiles to the base in order to keep the turret upright.

Figure 2. The PhantomX Pan Tilt Stand that forms the base of the ONT, with two installed AX-12 servomotors.[3]

Firing Mechanism

The ONT began as a Nerf Barricade RV-10, a 10-round semi-automatic blaster powered by three AA batteries. In its original configuration, the Barricade RV-10 used a manual on/off switch to power the two flywheels that propel the darts. With the flywheels spinning, the operator pulls the trigger, which advances a single dart forward through a set of mechanical linkages. After the dart fires and the operator releases the trigger, the barrel rotates to place a new dart in the proper firing position.

We wanted the ONT to repel adversaries from an office using repeated volleys of darts and thus needed to automate the firing mechanism. We considered two possible solutions to automate the ONT: either a servo to actuate the trigger (essentially a linear actuator pulling the trigger and releasing), which would use much of the original Nerf hardware, or a custom design solution that removes the use of a pull-back trigger altogether. However, we could not find a linear actuator capable of providing sufficient force to pull the trigger using our servos. Consequently, we decided to replace the barrel-fed design with a magazine-fed design.

We redesigned the firing mechanism to feed a single dart into the flywheels using a crank slider mechanism. After the crank slider system pushes the first dart into the flywheel system, the slider retracts and the next dart drops into the firing position waiting for the subsequent crank slider cycle; the finalized 3D-printed part appears in Figure 3. We

3 Image from: https://www.trossenrobotics.com/phantomx-pan-tilt

found a crank slider mechanism model and adapted its features to serve our needs.[4] Figure 4 shows the crank-slider mechanism and its relationship to other components. The modifications to the existing model include a pusher bar (green) that extends out from the mechanism. We attached a second, non- rotating slider pin adjacent to the existing slider pin (red). We also changed the overall dimensions of the slide (black) to accommodate the extra slider pin.[5] The remaining pieces – the wheel (white), lever (cyan), crank (magenta), and other hardware (red) – remain largely unmodified from the original design.

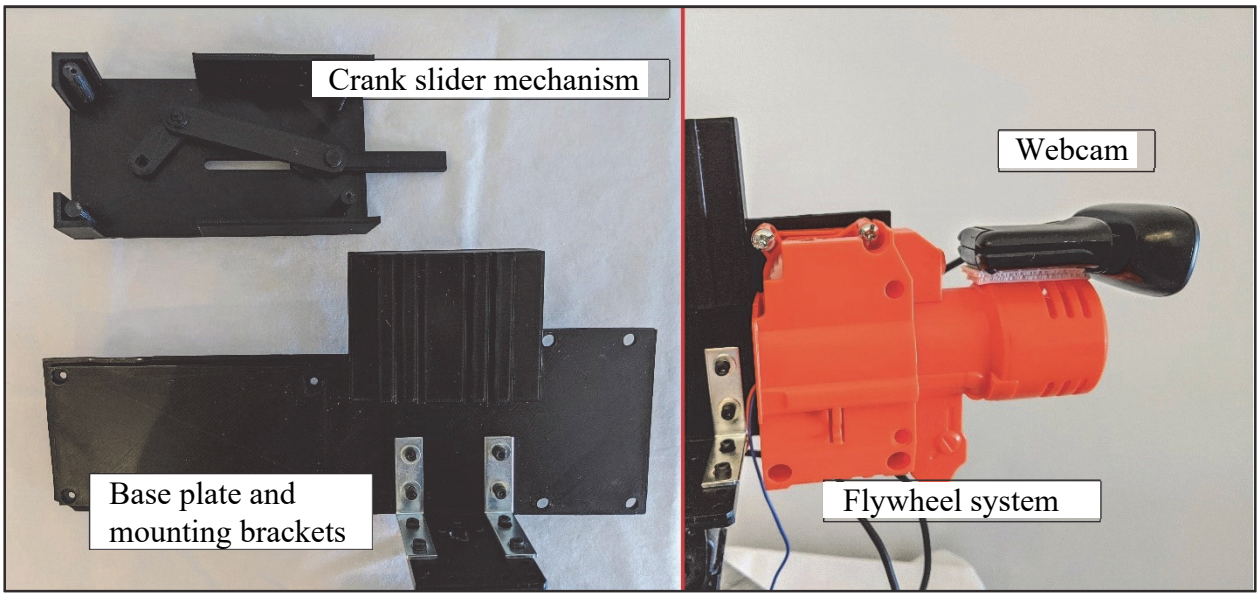

Figure 3. Internal view of firing components (left) and flywheel and webcam (right)

Figure 4 also shows a 3D model of the flywheel system (orange), a Nerf dart (dark blue), and the magazine (transparent, light gray). When the pusher bar fully extends, the dart advances into the flywheel system and the counter-rotating flywheels (shown on the right of Figure 4, located inside flywheel system) accelerate the dart out of the barrel. Once the pusher bar fully retracts, a new dart feeds into the chamber from the magazine above. An SG90 servomotor controls the wheel rotation from [0°, 180°] at a rate of about 30 rounds per minute. The current magazine capacity is 12 darts.

[4] https://www.thingiverse.com/thing:1241789

[5] The slide shows the restricted movement of the pusher arm and is not a separate piece of the firing mechanism base (see Figure 5).

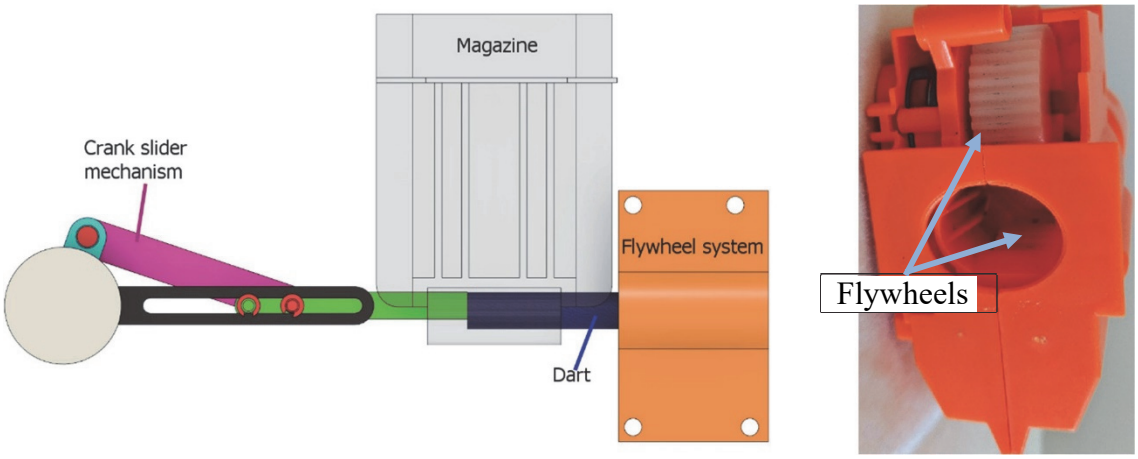

Figure 4. Left: Firing system, including dart (dark blue), flywheel system (orange), and magazine (transparent). Right: Interior view of the flywheel system showing the two flywheels (white).

Figure 5 shows the complete firing component assembly. Two small metal angle brackets (visible in Figure 3) hold the entire firing component assembly shown in Figure 5 to the pan-tilt base. The magazine clip (gray) secures the magazine to the turret, while the wheel (white) provides the connection from the SG90 servo (which controls the dart feeding process) to the crank slider mechanism.

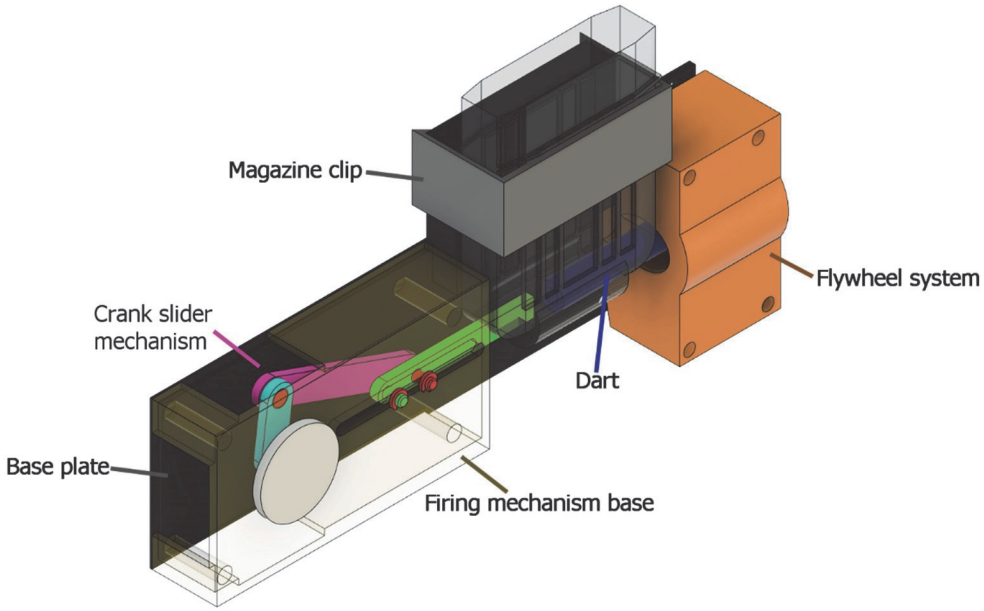

Figure 5. Complete firing component assembly (base plate (black), magazine clip (gray), firing mechanism base (transparent yellow)), dart (dark blue), flywheel system (orange), and crank slider mechanism (inside the firing mechanism base; see Figure 4 for color references)

Control Components

The ONT control components are responsible for observing the environment, running the software that allows the ONT to detect and identify faces, and sending commands to move the pan-tilt base and engage the firing mechanism. Figure 6 shows a high-level diagram of the control components and how they interface with the other parts of the turret.

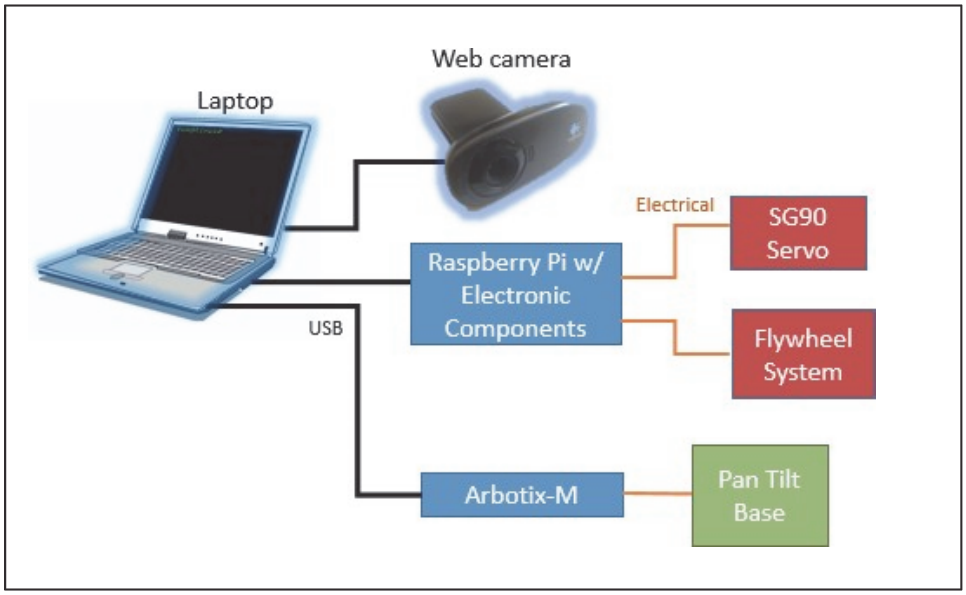

Figure 6. A high-level diagram of the control components of the ONT (in blue) and their connections to each other and other components of the ONT. USB connections are shown in black, while electrical connections are shown in orange.

A laptop using Linux Mint and two microcontrollers constitute the brains and control relay of the ONT. As discussed in the next section, the laptop handles the machine learning used to identify faces as well as calculations to command the pan-tilt to center targets in the field of view.

An HD 720p Logitech webcam connected to the control laptop and mounted to the turret barrel provides the turret's current field of view. The field of view of the camera aligns with the barrel, and as the pan-tilt moves, the field of view of the camera moves with the barrel.

The ArbotiX-M microcontroller serves as an intermediary to pass commands to the pan-tilt servomotors. Figure 7 shows a schematic of the microcontroller and servomotor electronic connections. The ArbotiX-M connects to the control laptop via a UART-to-USB cable. The two AX-12 servomotors in the pan-tilt assembly connect to the ArbotiX-M via their 3-pin servo header cables. Finally, a 12V power supply connects to the ArbotiX-M to provide power for the microcontroller and the two AX-12 servo motors.

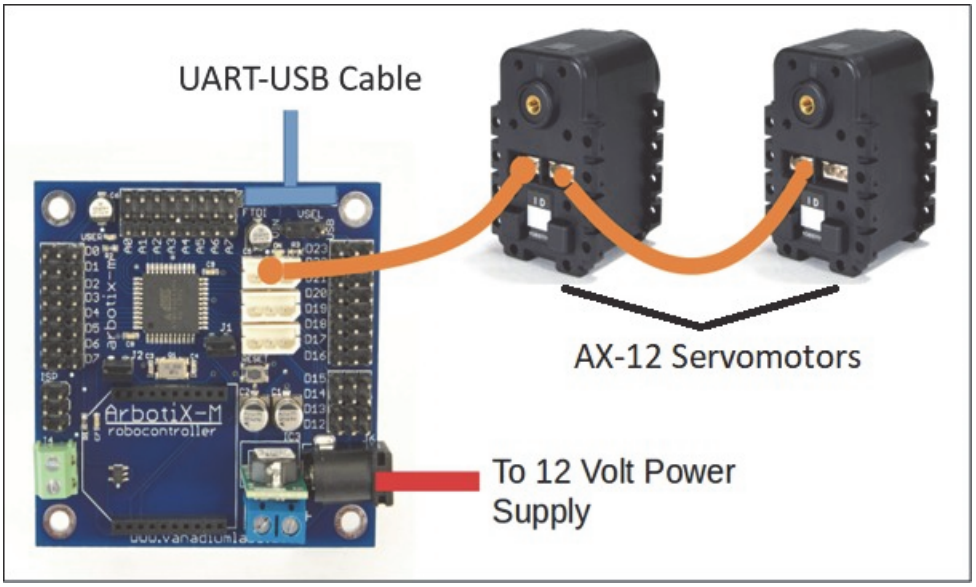

Figure 7. Wiring of the ArbotiX-M microcontroller to the AX-12 servomotors that control the pan and tilt positions of the turret[6]

A Raspberry Pi microcontroller, shown in Figure 8, relays commands from the laptop to the ONT's fire control mechanism via Micro USB.[7] Four of the 40 general purpose input-output (GPIO) pins on the Raspberry Pi are used to drive the fire control components of the ONT: (1) the 5V output pin and (2) the ground pin power the SG90 servomotor, (3) pin 17 relays the commands to the SG90 servomotor to turn the wheel and push darts into the flywheel, and (4) pin 27 triggers the flywheels to spin. When the laptop gives the command to "FIRE," the Raspberry Pi sets pin 27 to "high" (3.3V). The Raspberry Pi transmits this signal to the base pin of the 2N2222A transistor, allowing current to flow from the AA battery supply through the flywheels and transistor to ground, causing the flywheel to spin up. Three seconds later, the laptop commands the Raspberry Pi to actuate the SG90 servomotor via pin 17, causing the mechanical assembly to push a dart into the flywheels and launching the dart at the target.

The control circuit includes two other components designed to protect the system from voltage and current spikes. A 320 ohm resistor resides between pin 27 and the transistor to limit the current draw from the transistor. A diode sits in parallel with the flywheel load to prevent current spikes due to the flywheel's inductive load.[8]

---

6 Image adapted from: https://www.trossenrobotics.com/p/arbotix-robot-controller.aspx and https://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

7 Raspberry Pi Foundation, Raspberry Pi Zero W. https://www.raspberrypi.org/products/raspberry-pi-zero-w/

8 For more information, see: Wikipedia contributors. (January 28, 2020). Fly back diode. In Wikipedia, The Free Encyclopedia. Retrieved 01:31, March 11, 2020, from https://en.wikipedia.org/w/index.php?title=Flyback_diode&oldid=937955826

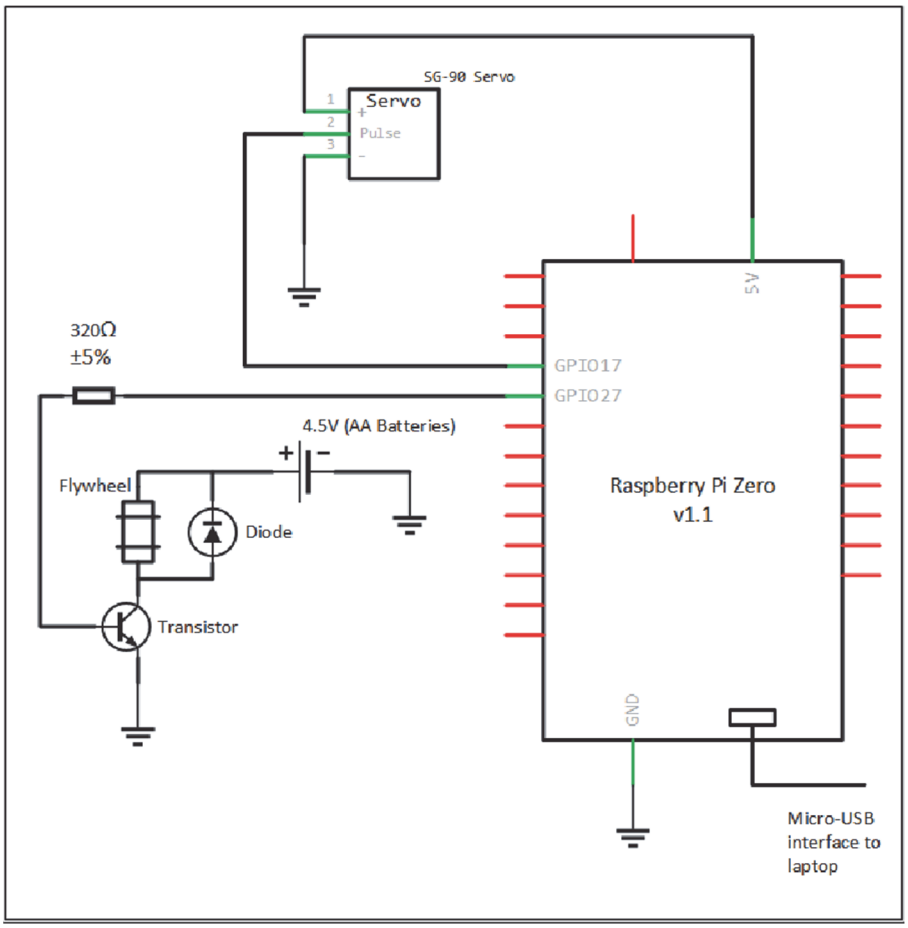

Figure 8. Wiring of the Raspberry Pi microcontroller to the SG90 servomotor, 2N222A transistor, and flywheel mechanism that constitute the fire control group of the ONT

SOFTWARE

The ONT software is responsible for taking the input from the web camera and identifying faces, orienting the barrel towards a target, and commanding the firing mechanism to launch a Nerf dart. The turret is programmed primarily using Python. All Python code runs on the control laptop, while the Raspberry Pi and the ArbotiX-M microcontrollers act as a link from the laptop to the turret hardware.

We adapted code from tutorials at Adrian Rosebrock's PyImageSearch to identify faces from an image and to orient the barrel towards a target.[9] When the turret is on, five processes/functions run simultaneously, sharing the state of variables through a process manager. Figure 9 shows a top-level diagram detailing the interaction of these processes.

---

[9] Adrian Rosebrock, Pan/tilt face tracking with a Raspberry Pi and OpenCV, PyImageSearch, https://www.pyimagesearch.com/2019/04/01/pan-tilt-face-tracking-with-a-raspberry-pi-and-opencv/, accessed on March 16, 2020.

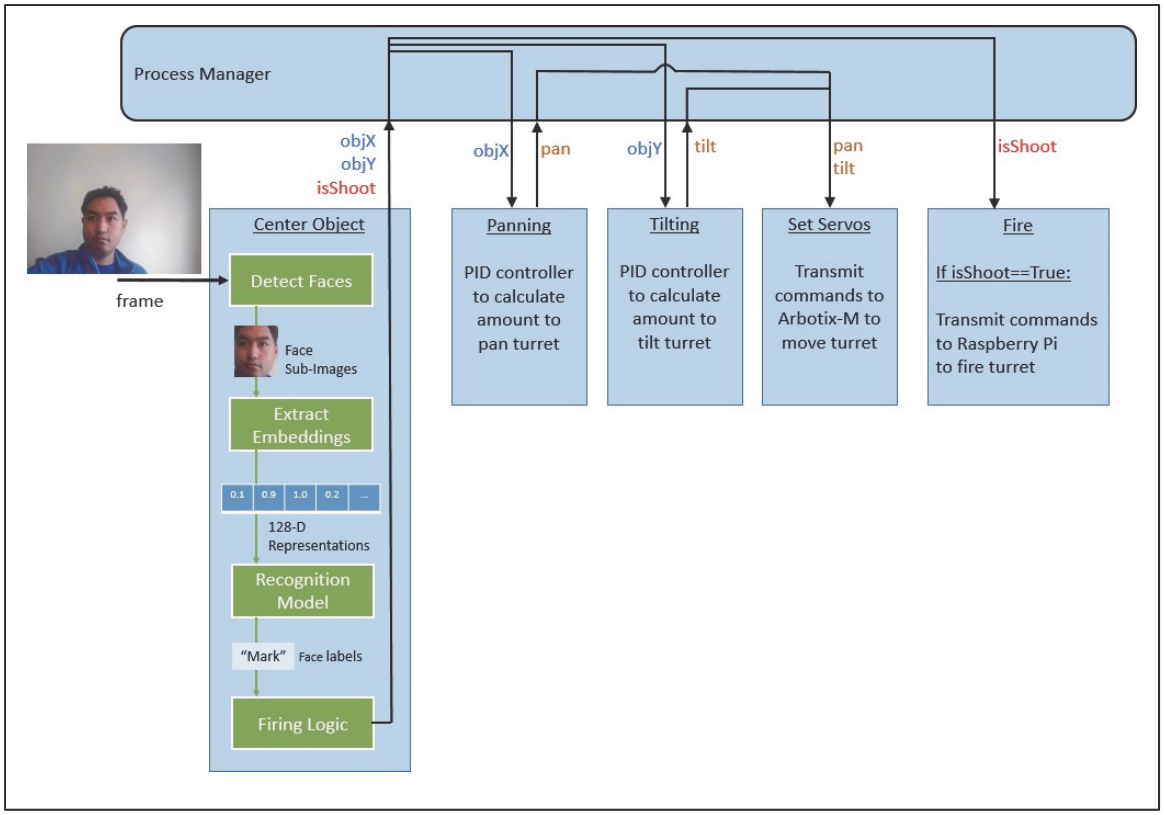

Figure 9. Top-level diagram of the flow of data implemented in the ONT hardware. Arrows indicate the input/output of variables between processes.

Center Object

The Center Object process is the most complex portion of our script, and is responsible for implementing a facial recognition pipeline to identify faces, as well as running through the firing logic to determine if the target should be engaged. It provides the coordinates of identified targets (`objX` and `objY`) as well as a fire flag (`isShoot`) to the other turret processes.

Facial Recognition Pipeline

Following the PyImageSearch tutorials, our facial recognition workflow follows a similar pipeline:[10,11]

[10] Adrian Rosebrock, OpenCV Face Recognition, PyImageSearch, https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/, accessed on March 16, 2020.

[11] Adrian Rosebrock, Face recognition with OpenCV, Python, and deep learning, PyImageSearch, https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/, accessed on March 16, 2020.

- **Detect Faces:** Our script imports a convolutional neural net tuned for facial recognition from the `openCV` Python library to find faces in the webcam's field of view (stored in the `frame` variable). The neural net computes a confidence score for each detected face. The script drops candidate faces that do not exceed a user-defined confidence score and candidate faces that do not pass some basic logical checks (e.g., adequate number of pixels).
- **Extract Embeddings:** The Center Object process imports a second convolutional neural net from the `facial_recognition` Python library that extracts a 128-dimensional representation of the detected face.[12,13]
- **Apply a Recognition Model:** A support vector machine (SVM) takes the 128-dimensional embedding of the faces and classifies them as either a known individual or an unknown face, based on a pre-defined "no-shoot/friendly" database.
  - **Training of the Recognition Model**: The SVM must first be trained prior to turret operation. First, we form a training set by combining pictures of various faces, with about 20 images for each labeled face. A training script extracts faces and their 128-dimensional representations for each image in our training set. Finally, we implement a supervised learning algorithm using the `scikit-learn` Python package to set the weights of our SVM to appropriately identify faces.[14]

Firing Logic

Once the faces in the field of view of the webcam are identified, the script follows a set of logical decisions on how to process the face:

- The Center Object process compares the names associated with identified faces to a list of "no-shoot/friendly" names contained in a user specified text file: `friend_database.txt`.
- If a face in the field of view is identified with a name contained in the friend database, it is ignored; the Center Object process does not return the location of the face for centering.
- If a face is identified with a name *not* in the friend database, the Center Object process flags that face and returns the location of the face (`objX, objY`) to the panning and tilting processes, centering the turret on the target.
- The video stream from the web camera highlights the hostile face in red on the user interface, as shown in Figure 10.

---

[12] Davis King, `Dlib` C++ Toolkit, http://dlib.net/, accessed on March 16, 2020.

[13] Adam Geitgey, `face_recognition` Python module, https://github.com/ageitgey/face_recognition, accessed on March 16, 2020.

[14] The sci-kit-learn developers, `scikit-learn` Python module, https://github.com/scikit-learn/scikit-learn, accessed on March 16, 2020.

Figure 10. The field of view of the webcam, demonstrating identification of friendly (green) and hostile (red) faces

- In order for the turret to activate and fire upon a hostile face, a valid target must remain in frame for a user-defined number of continuous frames (typically set to five). This prevents triggering of the target on anomalous or noisy output of the recognizer SVM. Once a target has been in frame for the requisite number of frames, the Center Object process sets the fire flag (`isShoot`) to "TRUE." This starts the fire process by powering up the flywheels to initiate the firing sequence.

One important design feature is that any changes to the `friend_database.txt` file are reflected in the turret logic in real time. That is, the set of no-shoot names are updated upon changes to the friend database without having to restart the turret script. This allows for real time integrity attacks against the database, increasing the system's utility as a demonstration tool.

Panning

The panning process accepts the current horizontal coordinate (`objX`) of a target. A proportional-integral-derivative (PID) controller (a common control loop mechanism) determines the position (`pan` variable) set by the servomotor controlling the pan of the turret in order to center the target horizontally in the camera field of view.

Tilting

The tilting process accepts the current vertical coordinate (`objY`) of a target. A PID controller determines the position (`tilt` variable) set by the servomotor controlling the tilt of the turret in order to vertically center the target in the camera field of view.

Set Servos

The set servos process takes the commanded `pan` and `tilt` positions from the panning and tilting processes and passes them to the turret servomotors via a serial interface to the ArbotiX-M microcontroller.

Fire

Monitors the state of the fire flag. If the flag is set to "TRUE," the process sets the Raspberry Pi's GPIO pins as described in the previous section, causing the turret to fire. The fire process also activates an audible alarm, providing a final warning of the turret's imminent activation.

COMMUNICATION PROTOCOLS

The ONT uses Python scripts to facilitate communication over two separate communication protocols: Transmission Control Protocol (TCP) and serial link.

First, the control laptop sends commands to the Raspberry Pi pins via the `gpiozero` library using TCP packets.[15] This implementation requires the control laptop and the Raspberry Pi to be on the same local area network, either via Ethernet, Wi-Fi, or Ethernet-over-USB. Remote GPIO connectionsmust also be allowed on the Raspberry Pi, and a service port (port 8888 by default) must be opened on the Raspberry Pi to receive commands from the control laptop.

The laptop provides serial commands to the two AX-12 servomotors that aim the turret camera and gun using the `pypose` library.[16] Prior to turret operation, we upload a standard `pypose` script file to the ArbotiX-M microcontroller, causing the microcontroller to act as a passthrough for the serial link to the AX-12 servos. This allows us to write commands and receive updates to the AX-12 servos via the laptop's serial bus.

Both the AX-12 serial connection and the TCP connection to control the Raspberry Pi GPIO pins are unauthenticated and unencrypted protocols, and serve as useful demonstration cyber-attack vectors into the ONT.

DESIGNED VULNERABILITIES

In line with its purpose as a cybersecurity demonstration tool, the ONT implementation contains a number of critical vulnerabilities that allow an adversarial agent to compromise the operation of the system.

First, there is no authentication or protection of the critical data file `friends_database.txt`. This plain text ASCII file is housed in the directory structure of the turret project, and is easily editable without requiring root-level access to

---

[15] Ben Nutall. `gpiozero` Python package. https://gpiozero.readthedocs.io/en/stable/index.html, accessed on March 16, 2020.

[16] Vanadium Labs. `pypose` Python package http://vanadiumlabs.github.io/ArbotiX/, accessed on March 16, 2020.

the system. Moreover, the implementation of the ONT provides no mechanism for checking the validity of the database file, such as signing and encrypting the database file. The weakness in this particular implementation of the database file is the focus of our DATAWorks mini-tutorial.

Second, both the TCP and serial connection to the Raspberry Pi and ArbotiX-M microcontrollers suffer from a lack of authentication and encryption. Thus, it would be straightforward for an adversarial agent to inject traffic along those interfaces to or from the turret. This would allow an adversary to inject commands such as moving servomotor position or initiating the fire sequence. The remote `gpiozero` interface can be conducted over a local area network. As a result, adversarial presence on that network (e.g., via physical access, Wi-Fi cracking, or remote exploitation) would allow an adversary to monitor all the packets to and from the turret, spoof or replay commands, or deny connectivity between the turret and the control laptop. These attack mechanisms are not the focus of our DATAWorks mini-tutorial, but will be highlighted in an upcoming IDA tutorial on man-in-the-middle attacks.

We intentionally made these design choices to help demonstrate cyber-attacks against the ONT, but insecure interfaces often appear in the real world. For example, insecure legacy protocols such as the HyperText Transfer Protocol (http) and the File Transfer Protocol are still common on the public internet. Insecure protocols can also be adopted to ensure backward capability with legacy systems, to minimize latency in a connection, or because their implementation is more straightforward. These vulnerabilities are intentionally built into the ONT and the system was designed without a focus on security. Consequently, other vulnerabilities are likely present in this software/hardware demonstration.

CONCLUSION AND POSSIBLE IMPROVEMENTS

The ONT is a dual purposed project. It serves as a tool to keep unauthorized individuals out of an area, and as a demonstration of the kinds of effects an adversary can achieve by exploiting a cybersecurity vulnerability. While leveraging machine learning, target recognition, and target tracking allowing the ONT to defend an area, these capabilities cannot be leveraged appropriately if an adversary compromises one of the many designed vulnerabilities in the system.

The ONT provides an adequate demonstration tool for the types of physical havoc an adversary can achieve with network compromise. However, changes can be made to improve the ONT as both a demonstration tool and an office privacy defender.

As a tool to deny access to unauthorized individuals:

- A more refined facial recognition training and processing pipeline, including adversarial examples, facial alignment, and high-speed graphical processor units to reduce latency.
- Redesign of the Nerf dart magazine feeding system and replace the SG90 servomotor to improve overall firing rate.
- Firing logic to deal with the presence of two or more hostile faces. Currently, the ONT can only process and prosecute a single hostile face in the webcam field of view.

- An improved webcam to improve the field and depth of view of the turret.
- A fly-by-wire remote control interface for the ONT, allowing the operator to control in real time the position and firing state of the turret, aided with target detection algorithms on the field of view.
- Replacing the insecure protocols implemented in the ONT with encrypted, authenticated protocols.

As a cyber security demonstration tool:

- Adopt other, non-Internet Protocol interfaces such as the Controller Area Network interface commonly used in automobiles.
- Add dormant, hidden parts of code (i.e., logic bombs) that trigger malicious effects based on certain input. For example, including obfuscated code that causes the ONT to shut down when a particular face is identified.
- Use common cyber defender/blue team tools (e.g., Nessus or Assured Compliance Assessment Solution scans) to identify additional vulnerabilities in the ONT implementation.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
April 2020	IDA Publication	

4. TITLE ANDSUBTITLE

Making an Office Nerf Turret: A Vulnerable, Cyber-Kinetic Demonstration

5a. CONTRACT NUMBER
Separate Contract

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
AE CRP Central Researh Project

6. AUTHOR(S)

Mark R. Herrera (OED); Jason R. Schlup (OED);

5d. PROJECT NUMBER
CRP

5e. TASK NUMBER
C9096

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

NS D-13158
H 2020-000135

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

10. SPONSOR/MONITOR'S ACRONYM(S)
AECRP

11. SPONSOR/MONITOR'S REPORT NUMBER

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for Public Release. Distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Our Cyber Lab team within the Operational Evaluation Division (OED) developed a mini-tutorial on basic cybersecurity concepts for the 2020 Defense and Aerospace Test and Analysis Workshop (DATAWorks) conference. To make the results of cyber compromise more tangible, we designed and constructed a desktop Office Nerf Turret (ONT) to be the centerpiece of the tutorial. This write up is a supplement to the mini-tutorial, and explains how we built the ONT and some of the vulnerabilities incorporated into the demonstration. This document lists the hardware used to build the turret, including the 3D printed firing mechanism. We also highlight useful tutorials that contributed to the development of the machine learning and tracking components of the ONT.

15. SUBJECT TERMS
Supervised Machine Learning; Computer Vision; cybersecurity; Kinetic Attacks; demonstration

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Peter Mancini (OED)
Unclassified	Unclassified	Unclassified	Unlimited	22	**19b. TELEPHONE NUMBER** *(include area code)* (703) 845-2496

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI std. Z39.18