­­Logo, company name

Description automatically generated­­­­A close-up of a circuit board

Description automatically generated with low confidence­­­­

Group 4

Goh Kheng Xi, Jevan (A0199806L)

Ryan Tang Li Yen (A0200900A)

Lock Mei Lin (A\*\*\*\*\*\*\*)

Datasheet

**C**LUSTER TRACKER

­­

**C**LUSTER TRACKER

The Cluster Tracker is designed to track clusters of humans through sensor fusion and artificial intelligence by obtaining metadata such as cluster size and distances between targets.

This system can be deployed to supplement that lack of manpower in enforcing social distancing rules in the light of the ongoing pandemic. Minor adjustments can be made to perform other tasks.

Contents

[1. Overview 3](#_Toc86553220)

[1.1 Components and setup 3](#_Toc86553221)

[1.2 General Specifications 4](#_Toc86553222)

[1.3 System Diagram 4](#_Toc86553223)

[2. Jetson Nano (Micro-controller) 5](#_Toc86553224)

[3. IWR6843 AOP EVM (mmWave RADAR) 6](#_Toc86553225)

[4. Intel RealSense L515 (3D Lidar Camera) 7](#_Toc86553226)

# 1. Overview

The Cluster Tracker is a system designed to track clusters of humans through sensor fusion. Its purpose is to alleviate the manpower required to enforce social-distancing rules through sensor fusion and artificial intelligence. Targets’ metadata obtainable by the sensors onboard the system are:

1. Velocity of targets relative to the system
2. Acceleration of targets relative to the system
3. Range of targets relative to the system
4. Azimuth and elevation of targets relative to the system
5. Number of targets present in the Field-Of-View (FOV) of the system

## 1.1 Components and setup

Key Components:

1. IWR6843AOPEVM from Texas Instrument
2. Intel RealSense L515
3. Host PC
4. NVIDIA Jetson Nano
5. MG995 Servo Motors

Other components:

1. MicroSD card (minimum 32Gb) x1
2. 4V 4A power supply (barrel jack) x1
3. Power supply for host PC x1

4. USB-A to micro USB-B cable x3

Key components 1 and 2 are the 2 sensors incorporated into the systems where IWR6843 is a mmWave/millimetre wave RADAR (see section 4) while L515 is a 3D Lidar Camera (see section 5). Figure 1 below shows how the setup looks like.

A picture containing diagram

Description automatically generated

5. Pan-tilt servo motors

4. Jetson Nano

1. IWR6843 AOP EVM

Figure 1. Cluster Tracker Setup

2. Intel RealSense

L515

3. Host PC

Mounts

## General Specifications

|  |  |
| --- | --- |
| FOV (horizontal) |  |
| Range | 12 m |
| Maximum velocity | 6 m/s |
| Max number of targets | 13 |
| Accuracy\* | * To be tested - |

This section describes the general specifications of the Cluster Tracker (Table 1) obtained from testing of the system. The tests are conducted in an indoor environment of about 12m by 8m area, under normal room lightings and about .

\*Refers to the average percentage number of times social distancing rules are broken without being detected by the Cluster Tracker over the total number of times social distancing rules are broken

Table 1. General Specifications

## 1.3 System Diagram

Figure 2 shows the system architecture of the Cluster Tracker and the operational flow.

Diagram, timeline

Description automatically generated

Figure 2. Diagram of the system architecture and operational flow

Due to the complexity of the system, the operational flow described in Figure 2 can be parsed into the following 3 major components:

RED: IWR6843 (mmWave Radar sensor)

Green: Jetson Nano (microcontroller) and servo motors

Blue: L515 (3D camera)

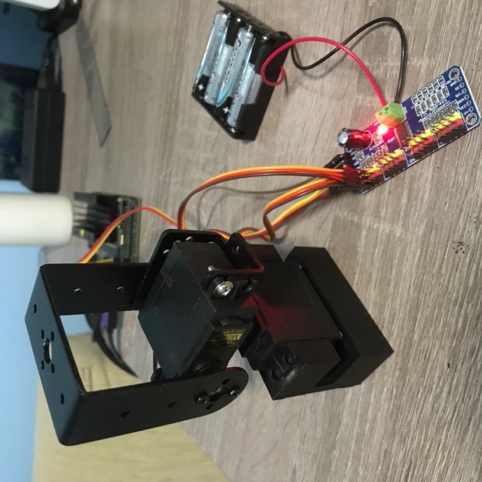
All 3 components interfaced with one another on the Host PC denoted by the white rectangular box.

# 2. Jetson Nano and Servo Control

|  |  |
| --- | --- |
| Power to servo motor | 5 V from 4 x AA alkaline cells |
| Logic level | 3.3 V |
| Actuation range | 120º |
| Pulse width range | 1000 – 2000 μs |
| Operation speed | 0.2 s / 60º (for 4.8 V) |
| Torque | 0.83 N m (8.5 kgf cm) |

The NVIDIA Jetson Nano (henceforth called Nano) controls the pan-tilt servo motors of the L515 3D camera such that we are able to track clusters of people across the full 120º azimuthal field of view of the IWR6843 mmWave sensor and have a greater lateral (x-y) view. The following figure and table shows how the L515 is mounted onto the servo motors and the specifications of the servo motors.

Figure 3. Assembly used for testing. The pan and tilt servo motors are connected to the PCA9685 PWM/servo driver.



133 mm

61 mm

60 mm

Table 2. Specifications of the MG995 servo motors.

The PCA9685 servo driver connects via I2C to the Nano which uses 3.3 V logic for the pulse width modulation (PWM). On board the Nano is the CircuitPython library by Adafruit that controls the servo motors; it simplifies the PWM with its built-in module that converts the angle inputs (in Python) to pulse widths for the servo motors.

Now the Nano serves only as a microcontroller given that the software development kit (SDK) for the mmWave, built in a 32-bit x86 architecture, is incompatible with the 64-bit ‘aarch64’ ARM CPU; object detection and tracking by the L515 is also done on a PC for the time being. Therefore, we use PuTTY to interface serially with the Nano; this, however, posed difficulties for data transfer between the servo motor, mmWave, and L515 scripts. Some attempts were made to set up parallel processing (and hence data sharing) via the Python ‘Multiprocessing’ package, but using ‘Socket’ programming lent a faster solution. Here the Nano is set up as a server ‘socket’ that establishes a wireless or LAN connection via the Transmission Control Protocol (TCP) with the PC (client ‘socket’), enabling real-time interfacing between the devices with negligible latency. For instance, the full process - data transfer from the mmWave to the Nano and panning the L515 90º - only takes around a second.

# 3. IWR6843 AOP EVM (mmWave RADAR)

The IWR6843 AOP EVM is a millimeterwave RADAR that operates on gigahertz frequencies. Due to the relatively higher frequency and thus, shorter wavelength compared to other RADARs, it has a higher accuracy but lower range.

## 3.1 Specifications

The IWR6843 AOP EVM is tested in various indoor environments with normal room conditions including a 20m by 8m laboratory environment and an 8m by 5m homeroom environment. The specifications are based on the tests conducted.

|  |  |
| --- | --- |
| Frequency | 60 GHz |
| FOV (horizontal) |  |
| Range | 16 m |
| Maximum velocity | 6 m/s |
| Minimum velocity | 0.032 m/s |
| Max number of targets | 6 |
| Resolution\* | 4mm |
| Accuracy\*\* | ~ 80% (50% to 90%) |

\* Refers to the minimum distance that two data points can represent (i.e. surface area of targets divided by number of data points)

\*\* Refers to the ratio of range readings of target by sensor to the actual range of target

Maximum number of targets tracked by the IWR6843 is low due to RADAR not being capable to distinguish separate targets in proximity of one another and hence, the need for L515. However, even when targets are not in close proximity, the maximum number of targets tracked by the IWR6843 is found to be 6 which could possibly be due to the configuration of the sensors. Additionally, the sensor might lose track of the targets occasionally and cause abnormal drops in accuracy.

Work is being done to improve the quantity and accuracy of point cloud data through the incorporation of another DCA1000 board by Texas Instrument that allows the reading of raw data input.

## 3.2 Signal specifications

The IWR6843 operates using frequency-modulated continuous wave (FMCW) of approximately 60 – 64 GHz, which is unlike conventional radar system that transmits short pulses at regular intervals. Due to this, the IWR6843 can obtain the following information from the transmitted and received signals:

1. Range of target
2. Velocity of target
3. Acceleration of target
4. Angle of Arrival (AoA) of signal which corresponds to angle and elevation of target

## 3.3 Serial output

The data output of the IWR6843 is dependent on the image flashed onto the hardware. For this Cluster Tracker, the IWR6843 outputs the serial data through UART (Universal Asynchronous Receiver Transmitter) which is read by the Host PC through a USB port installed with a USB to UART driver. The serial output can be read and parsed using the pyserial library from python.

The serial output are of little endian byte order (Figure x) and are TLV(type/tag-length-value) encoded. The encodings for the packets are described below.

**Frame Header (48 bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Synchronization Pattern | Uint64 | 8 | “02 01 04 03 06 05 08 07” |
| Version | Uint32 | 4 | Version of SDK |
| Total Packet Length | Uint32 | 4 | Inclusive of Header |
| Platform | Uint32 | 4 | 0xA6843 |
| Frame Number | Uint32 | 4 | - |
| Sub Frame Number | Uint32 | 4 | - |
| Chirp Processing Margin | Uint32 | 4 | - |
| Frame Processing Margin | Uint32 | 4 | - |
| Track Processing Time | Uint32 | 4 | - |
| UART Sent Time | Uint32 | 4 | - |
| Number of TLVs | Uint16 | 2 | - |
| Check Sum | Uint16 | 2 | Header check sum |

For each TLV payload, there is a header which is encoded as follows:

**TLV Header (8 bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Type | Uint32 | 4 | “06”, “07” or “08” |
| Length | Uint32 | 4 | Length of current TLV inclusive of TLV header |

There are 3 types of TLV payloads and they are:

|  |  |
| --- | --- |
| Type Number | Type |
| 06 | Point cloud |
| 07 | Target object list |
| 08 | Target index |

**Point Cloud TLV (Type 06)**

For point cloud TLVs, the first section of this payload is a structure consisting of the units to be multiplied to each of the point’s attributes. This is so that data can be compressed into lesser bytes. The point unit structure encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Elevation Unit | Float | 4 | - |
| Azimuth Unit | Float | 4 | - |
| Doppler Unit | Float | 4 | - |
| Range Unit | Float | 4 | - |
| SNR Unit | Float | 4 | - |

The rest of this TLV payload are an array of point structure of 8 bytes each and the encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| Elevation | Int8\_t | 1 | - |
| Azimuth | Int8\_t | 1 | - |
| Doppler | Int16\_t | 2 | - |
| Range | Uint16\_t | 2 | - |
| SNR | Uint\_16 | 2 | - |

**Target List TLV (Type 07)**

Target Object List TLVs contains an array of targets and their respective metadata. Each target are encoded in 112 bytes and the encodings are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| ID | Uint32 | 4 | ID of target |
| x-coordinate | Float | 4 | - |
| y-coordinate | Float | 4 | - |
| z-coordinate | Float | 4 | - |
| x-velocity | Float | 4 | - |
| y-velocity | Float | 4 | - |
| z-velocity | Float | 4 | - |
| x-acceleration | Float | 4 | - |
| y-acceleration | Float | 4 | - |
| z-acceleration | Float | 4 | - |
| Ec[16] | Float | 16 x 4 | 4 x 4 error covariance matrix (range, azimuth, elevation, doppler) |
| Gating function gain | Float | 4 | - |
| Confidence level | Float | 4 | - |

**Target Index TLV (Type 08)**

Target Index TLV consist of a list of target ID of 1 byte each and are encoded as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Value Type | Length (bytes) | Value/additional comments |
| ID | Uint32 | 4 | ID of target |

4. Intel RealSense L515 (3D LiDAR Camera)

The Intel RealSense L515 operates is a time-of-flight depth sensing 3D camera that uses an infrared light to collect depth data. The L515 is 61mm in diameter and 26mm in height. It weighs about 100g, and the power required is 3.5W, which is much lower than the other TOF depth sensors.

## 4.1 Specifications

The specifications of L515 are based on the experiments conducted in the laboratory, as well as in my house in low ambient lighting.

|  |  |
| --- | --- |
| Depth sensor | |
| Range | 5.6m |
| Horizontal FOV | 70° |
| Accuracy\*\* | ~ 93.87% |

|  |  |
| --- | --- |
| RGB sensor | |
| Range | 16m |
| Horizontal FOV | 55.8° |
| Max number of targets | 21 |

\*\* Refers to the ratio of range readings of target by sensor to the actual range of target

## 4.2 Artificial intelligence

The object detection algorithm used to identify human in the L515 video stream is the Haar cascade, which uses line detection features to identify human faces. The classifier is trained with images that contain faces, as well as images that do not contain any faces. The trained model is available from OpenCV.

In combination with the IntelRealSense python library, the distance between the human and the camera can be obtained, as well as the distance between each human detected.