MQTT Die Handoff Project Code Description

Lacey Bowden, Michael Habermann
CS 5553
Fall 2025

Code Overview

Lacey controls Bill (robot A), so her code is responsible for the initial die pickup, first MQTT packet publication, timing the program, and keeping track of the program loops. Michael controls DJ (robot B), so his program waits for the first message, converts the coordinate in the packet, sends the translated coordinate and gripper commands to DJ to begin the handoff sequence. One program loop consists of: Bill finding random coordinates, moving to them, publishing them, and subscribing to DJ; DJ grabbing die, publishing gripper status, subscribing to Bill; Bill releasing gripper, moving to a safe location (outside the possible handoff space), publishing gripper status, and subscribing to DJ; DJ moving to a random location, publishing coordinates, and subscribing to Bill; Bill translating and moving to coordinates, grabbing die, publishing gripper status, and subscribing to DJ; DJ releasing, moving to a safe location, publishing gripper status, and subscribing to Bill; and finally Bill moving to a safe location, and publishing a new set of coordinates (to start another loop), or publishing a packet with the 'loop_finished' variable set to True, indicating the program is over. If it is over, DJ will wave bye and return home, and Bill will place down the die in its start location and return home.

Robot Control

We utilized the FANUC Ethernet/IP Driver to control each robot and its corresponding gripper.

JSON Formatting

Encoding

Both of us had to convert a dictionary to a JSON before publishing to the broker, done in the function 'package_json()'. This function accepts the values 'coordinate' (a six-value coordinate list); 'gripper_status' (a Boolean with True indicating closed gripper, False indicating Open); and 'loop_finished' (a Boolean with 0 indicating program is still running and 1 indicating the program is done). The code later utilizes offsets to convert passed coordinates to its respective robot coordinates (as defined in a later section). The variable loop_finished is only changed to True by Lacey, as she (Bill, robot A) is counting loops and Michael (DJ, robot B) checks for this end condition frequently to know when to stop. These parameters are assigned to their respective dictionary keys in a dict called 'data' before being converted to a json object using Json.dumps() and returns the encoded message.

Decoding

For decoding json objects, we utilized the function 'parse_msg()' (Michael's code), or 'decode_json()' (Lacey's code), which essentially do the same thing. A json object is passed as a parameter, converted to a dictionary using json.loads(), and returned.

MQTT

Broker

Michael ran the Mosquitto broker on his local machine and this was used to handle all packet communication and topics.

Publish

The publish function publishes to a specific topic: DJ publishes to robot/dj and Bill publishes to robot/bill. These are handled by the aforementioned Mosquitto broker.

Subscribe

The ‘subscribe()’ function waits for the message to arrive and ‘on_message()’ defines behavior for packet reception. In Michael’s code (controlling DJ), he subscribes to the topic robot/bill and in Lacey’s (controlling Bill), she subscribes to robot/dj. After receiving the message, it is parsed and returned.

Coordinates

Generation

The function ‘find_random_position()’ accepts no parameters and returns a randomized position both robots can reach. It does this by adding a random offset between -150 and +150 to each cartesian value (x, y, and z) in the center location. This center location, called ‘handoff_middle_pos’ in both of our code files, is stored as a list and is the center of the robot’s reachable handoff space. The latter three values in the list are not set in this function because they are robot-specific and not necessary to be passed. This function is called every time one of the robots has possession of the dice (presumably, given their gripper is closed although we have no feedback yet) and needs to go to a new random location to pass to its robot friend.

Translation

Once a robot receives a coordinate, it must translate it from its robot friend’s coordinate system to its own. It does so by adding an offset to the y and z values. This offset was manually determined by us prior to coding by placing the robot arms on a die in the center of their shared reachable space and subtracting one position list from another. The offset value for y is 1750 (subtracted for Bill, added for DJ), and the value for z is 8 (added for Bill, subtracted for DJ). Michael did this conversion upon receiving an MQTT message (in his ‘parse_message()’ function, the equivalent of Lacey’s ‘decode_json()’ message), and Lacey did it using a function named ‘get_new_pos()’. Lacey’s function creates a new position list, accesses the global ‘message_i_got’ string (which is set in ‘on_message()’; assigns the ‘x_parameter’, ‘y_parameter’, and ‘z_parameter’ values to their corresponding coordinate indices (0, 1, and 2, respectively); and assigns the ‘yaw’, ‘pitch’, and ‘roll’ values to their corresponding indices (3, 4, and 5, respectively) before adding the offsets and returning the translated coordinate value.