# Computer - Arduino Communication Protocol

This document describes the communication between the fingerprint robot and the computer.

1. Communication
   1. The Arduino will receive input from a serial stream via USB or over a network connection on port 80 via Ethernet. The format of the requests will be the same in both cases.
   2. When the Arduino is connected over the network, it is capable of receiving requests over both a UDP connection and a TCP connection. However, if any request is received over a UDP connection, the response message will be send over a TCP connection.
2. Verification
   1. There may be many devices on the network and connected via USB. As such, the Arduino must be identifiable. When a computer connects to a device, the computer should transmit the string "fingerrobot" to the device. If the device is indeed the fingerprint robot, it should respond with the string "youfoundme" and thus confirm its identity. This should allow a computer to easily determine which device is the Arduino.
3. Communication
   1. Moves
      1. A move describes the motion of one servo motor. There are two types of moves: position moves and sensor moves.
         1. Position Moves
            1. A position move is when a servo motor moves to a specified position. Each position move consists of two parameters: the target servo position and the time allotment. For a position move, a command is sent to the servo to move to the specified position. During the time allotment, which starts as soon as the Arduino sends the command to the servo, no other commands will be sent to the servo.
            2. Format: pos <absolute position (degrees)> <time allotment (ms)>
         2. Sensor Moves
            1. A sensor move is when a servo motor moves until its sensor readings indicate that it is finished. Each sensor moves consists of 5 parameters: the servo’s angular velocity, the minimum light threshold, the maximum light threshold, the position threshold, and the wait time. For a sensor move, the servo will rotate at the specified speed until sensors indicate that the move is complete. Then, the servo will wait for the specified time. The sensors will indicate that the move is complete when the servo’s target position goes past the position threshold (if the servo is rotating forwards, it will stop when its position is greater than the position threshold, if the servo is rotating backwards, it will stop when its position is less than the position threshold) or the light sensor reading is between the maximum and minimum values.
            2. Once the waiting period has completed, the Arduino will return one of two possible values. If the sensor move completed because the light sensor reading moved into an acceptable rage, the Arduino will return a string of the following format: finger-<finger number>-successful-<force>. If the move finishes because the servo the position threshold is exceeded, the Arduino will return a string of the following format: finger-<finger number>-failed-<force>.

The force returned will be the value from the force sensor when the move finishes.

* + - * 1. Format: sen <speed (degrees/second, a negative value means the servo will rotate backwards)> <lower light threshold, inclusive, in the range [0, 1023]> <upper light threshold, inclusive, in the range [0, 1023]> <position threshold (degrees)> <wait time (ms)>
  1. Finger Moves
     1. A finger move is two servo moves; one move that places the finger down onto the fingerprint scanner and one move that lifts the finger off the scanner.
     2. The finger move data is stored in a string. The finger move is simply the concatenation of two servo moves.
  2. Command
     1. The following describes the command set that the Arduino will use.
     2. Set command
        1. The set command causes the Arduino to set move data for 1 servo.
        2. Format: set <servo number> <finger move>
        3. The servo number will be in the range [0, 3]. The move will be in the notation described in section 2.2.2. This method returns the string “set-end” upon completion of the move.
     3. Move command
        1. The move command causes the Arduino to execute the stored move data, for specified servos.
        2. Format: move <number of servos> <servo ...>
        3. The command takes a list of space-separated servo numbers and executes their moves synchronously. The number of servos should be the first argument. If the first argument is 0, every servo’s move is executed. For example, move 2 1 2 would move servos 1 and 2. move 0 would move all the servos. This method returns the string “move-end” upon completion of the move.
     4. Get command
        1. The get command returns a sensor value. The command can get the value of any of the 4 FSRs and any of the 4 LDRs.
        2. Format: get <servo> <sensor>
        3. The command takes two parameters. First, the servo number that is associated with the desired sensor. Second, the desired sensor type, either “f” or “l”. For example, get 3 f would get the FSR value for the FSR on servo 3. The value will be returned as a string, over the stream that the computer is using to communicate with the robot.
     5. Relax command
        1. Relaxes the servos so that the fingerprints can be changed.
        2. Format: relax
     6. Hold command
        1. Puts the servos back to their original positions.
        2. Format: hold
  3. Once the Arduino has received and started the execution of a command, it will return a string in the following format: “<command name>-received” and then proceed to execute the command. Once the received message has been sent, the Arduino will be able to receive another command. However, if two commands are sent in close succession, and the second command arrives before the Arduino has sent the received message for the first command, the second command may be lost due a buffer overflow on the Arduino. When connected over USB, the Arduino has a very small 64 byte buffer, making an overflow likely. However, when connected over Ethernet, the Arduino has a 16K byte buffer, thus making an overflow less likely.
  4. Once the Arduino has finished a command, it will return a string in the following format: “<command name>-end”.
  5. If the Arduino receives a bad command, it will return the following string: “bad-command”.