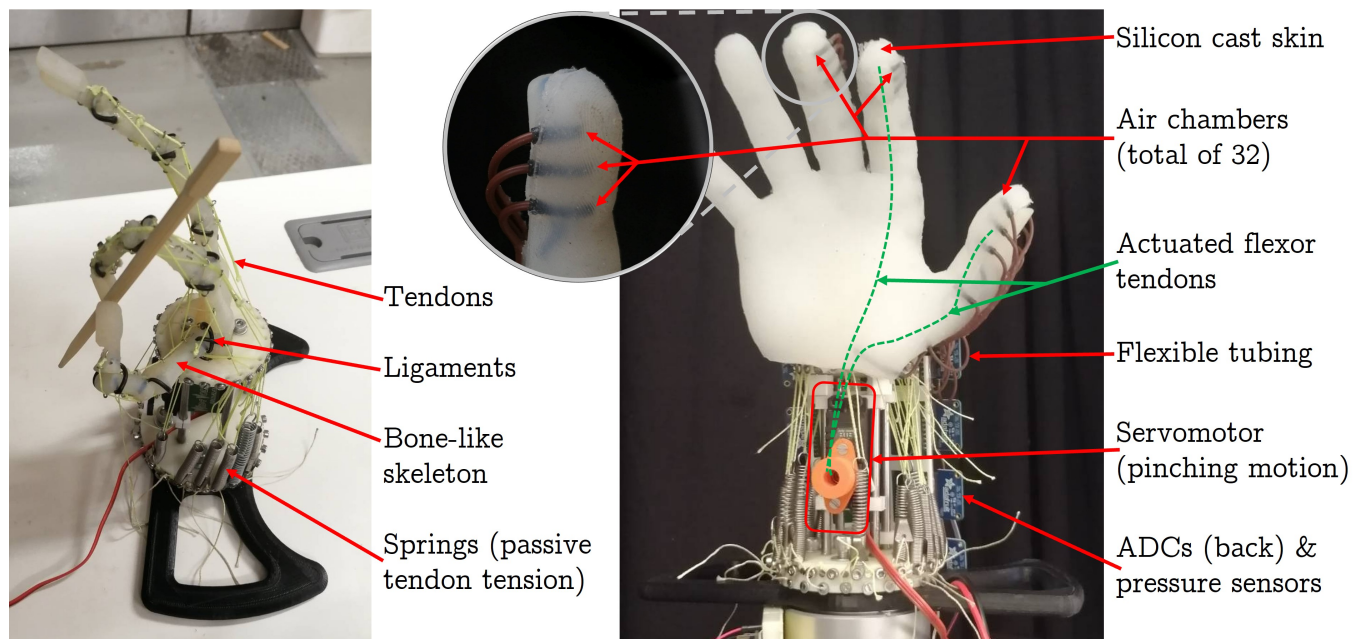


# IIB Project: A soft sensorised anthropomorphic hand for improved robotic manipulation



This documentation is provided as a complement to the Master's thesis submitted. It contains a description of the main software used in this project. The software and relevant data has been uploaded to the following [repo](#) which contains a formatted duplicate of this README.

## Table of contents:

- **Main controllers:** This folder includes all of the software used to control the UR5 robotic arm, the hand servomotor and the read the sensors' data. It also includes the data from all the performed experiments as well as any code used for signal processing or visualising the results.
- **Neural networks:** This folder contains the code implementations and results of the neural networks used for data interpretation of the surface orientation prediction experiment.
- **CAD:** This includes the CAD models generated for any part that required 3D printing or casting during the various experiments. The CAD of the hand and skin itself isn't included, its Kieran Gilday proprietary resources.
- **Media:** A few images and videos of the robots, experimental setups and results to showcase the project's system.

## Main controller

This section describes the controller used for experiments and data collection in this project. It is based off previous work done in the Bio-Inspired Robotics Lab ([BIRL](#)), notably PhD student Kieran Gilday. This [GitHub repo](#) contains the generic code that was used as a base for the UR5 controller and client software. For details on how the connection with the UR5 is setup, how to load the program on the robot and setup IP ports adequately, please refer to the aforementioned [GitHub repo](#).

## Contents

1. The Main-controller folder contains [ur5\\_and\\_hand\\_controller](#) which is the directory containing all controllers, collected data and function used for plotting. The data, the plotting functions as well as the functions used to collect said data are all named appropriately and briefly commented on in the code.
2. It also contains [Arduino](#), which is all the code used for sensor polling and servomotor control. This code should be uploaded to an Arduino or equivalent following the indicated pin numbering to allow for a controller to communicate with it.
3. This folder also contains [ur5\\_client](#), the code used on the UR5 client which is required to interface with the python script. This should be loaded on the UR5 system previously to any connection, see below.

### Libraries in use:

- numpy
- matplotlib
- csv
- cv2
- json
- math
- time
- scipy
- random
- pathlib
- \_thread
- socket

### Getting Started:

- Main loop in [Generic\\_ur5\\_controller.py](#), contains examples of using kg\_robot and teach mode. The modified implementation used in the thesis is presented in [sensing\\_hand.py](#).
- The robot is controlled using the functions defined in [kg\\_robot.py](#) in the ur5 commands section, if the desired robot function doesn't exist and cannot be created through a combination of existing functions, it can be added by modifying kg\_robot.py and kg\_client.urp. Complete capabilities are described in [ur5 script api](#).
- Specialized robot functions and complex sequences which include robot motion, cameras, hand actuation, data recording are called from [sensing\\_hand.py](#) and defined in [skin\\_sense.py](#)
- [Waypoints.py](#) is used for global robot poses, joints and tool centre points (tcp) which can be called by any function.
- To improve safety and ensure that the robot doesn't move rapidly to a previously defined position that may now be obstructed, a check is implemented at the start of [sensing\\_hand.py](#), as soon as the connection with the UR5 is completed, to assert the location of the new home position and redefine it as the current position if required.

## Neural Networks

This section relies on a set of iPython notebook based on a tutorial from [machinelearningmastery](#). The [Neural-networks](#) contains the code used for orientation predictions using an ANN and LSTM network, the data used to train and test these networks, and the obtained results. Folders containing results for each of these implementations are labelled appropriately.

## Contents

1. The folders [orientation\\_prediction\\_ANN](#) and [orientation\\_prediction\\_LSTM](#) each contain a .ipynb file which describe the entire training process and results for each network type. The folders also include the data used for training and testing as well as the some images showing training loss and network outputs.
2. A comparison of both results is included in [ANN-LSTM-comparison.ipynb](#)
3. Graphs showing results from varying networks and training parameters are included in [media-results](#).
4. An experiment was attempted to predict the 2D orientation of a pinched chopstick with the bottom end of the chopstick stuck in a fixed ball joint using a similar neural network data-driven approach. Results were inconclusive and disregarded in the thesis but data and training methods are included in [ball\\_joint](#)

## Libraries in use:

- Sklearn
- Keras
- TensorFlow
- Numpy
- Pandas
- Matplotlib
- Csv

## Getting Started:

The implementations follow the following structure:

1. Importing the data from a .csv file
2. Processing the raw data to extract suitable dataset and format it appropriately for use in neural networks
3. Splitting the data in training and testing datasets with shuffle if required
4. Define model topology
5. Perform training and plot loss in real time
6. Print an example of prediction from the testing dataset

These steps can be followed to understand the methods used and replicate these techniques for similar problems.