Project Proposal

Human Interface for Robotic Control

SYSC 4907 - 4th Year Project

|  |  |
| --- | --- |
| Brandon To | 100874049 |
| Minh Mai | 100845949 |
| Yuzhou Liu | 100853392 |

Team Members:

Supervised by: Richard Dansereau

September 18th 2015

Carleton University

# Introduction

This document is a proposal for the SYSC 4907 fourth year project, under supervisor Dr. Richard Dansereau. The project will last from September 2015 to April 2016. The team consists of 3 members: Yuzhou Liu, Minh Mai, and Brandon To.

# Objective

The project aims to develop a robot arm system that mimics human movement and remote control over the internet.

The project aims to develop an interface between intuitive natural human movements and robotic control. Specifically the target objective is for a robotic hand to mimic the movements of a human hand with minimal latency and high accuracy. The system is to be developed such that a user at a remote location can control the robotic hand over an internet connection.

The system is a low-cost development platform for robotic applications involving hand and arm control. The intent is to showcase current robotic technology that may find applications where human interaction is necessary but human presence is unsafe or inconvenient. Such remote control applications may include bomb diffusion, and assembly line work for tasks such as welding and parts placement.

# Background

Robotics has long been an area pursued by industry and hobbyists. Its applications today range from manufacturing plant assembly lines, to self-driving cars, to kid’s toys and beyond. Previously this area of study required a high initial cost for equipment and long lead time to design and manufacture the hardware. This led it to only being a specialized topic of study to those with enough funding and time available. However now with the practicality of 3D printers, the hardware can be obtained as simply as downloading open-source designs and printing it. One such open-source project is InMoov which is a complete life size human robot replica that is put together using 3D printed parts. With InMoov, the inclusion of robotics hardware into the 4th year project became practical.

The vision of this project is to develop a true embedded robotics platform that can serve as the foundations for developing robotic applications that replicate human movements. Due to the academic background of the 3 team members in electrical and computer systems engineering, it was decided that the system should have very close control to the hardware. For this requirement a hobbyist microcontroller such as the Arduino Uno was unacceptable. A combination of a Texas Instruments Tiva C Series microcontroller and Leap Motion for interfacing user interaction was the best possible choice for the goals outlined. Features such as remote control over internet and record/playback of user action were decided to be added to the platform to widen the scope of future development opportunities.

# Project Description

In this project, the motion sensor captures the motions of the human arms and send the data to the computer. The software in the computer processes the data and sends it over the internet to the micro- controller. At the other end, the micro- controller uses the data to mimic the human arm movements.

The hardware design consists of 3 main components: a hand motion sensor, a micro- controller, and a 3D – printed robot arms (see Figure 1). The hand motion sensor used in this project is the Leap Motion. A Texas Instrument CC3200 SimpleLink Wi-Fi LaunchPad is chosen as a micro-controller to control the robot arm. The InMoov designed robot arm is 3D printed and assembled to be used for the project. Each of the components are discussed below.

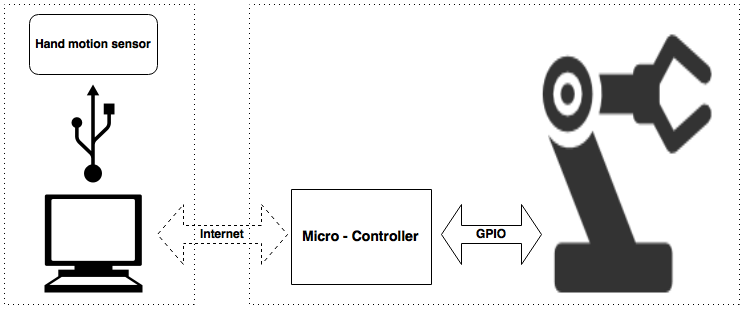


Figure 1: System overview

## 3.1 Leap Motion Sensor

The Leap Motion controller is a small USB peripheral device that provides accurate hand and finger tracking through the use of the sensor embedded in the device. The Leap provides libraries to query the device for several programming languages including C++, Python, Objective-C, and Javascript. For the purpose of our project, we will likely use C++ to interface our Computer with the Leap device. However, we would like to keep our options open, so we are not discounting Python.

The main challenge of the component in the project is identifying the relevant data that the Computer needs from the Leap Motion Controller. The Leap Motion provides tracking for the hand and each individual joint of the fingers, vectors representing the normals and direction in which they are facing, and the velocity in which these they move or rotate. Tracking all of this is a bit overkill for our application due to limitations in the movement of InMoov.

## 3.2 TI CC3200 SimpleLink Wi-Fi LaunchPad

The microprocessor controls the robot arm upon receive the relevant data from the Motions Sensor. Furthermore, it should calculate the speed at which the joints are moving, and will configure the PWM accordingly in order to simulate that movement on the servo motors.

The TI CC3200 WI-FI LaucnhPad is chosen because it provided a full support of the TCP/IP stack and Internet protocols which allows the relevant data from users to be received over the internet.

The main challenge of the component is that since the InMoov is driven by servo motors, the speed at which it can move will likely be much slower than the potential movement of our hand. We must handle the cases where the tracked hand moves faster than the servos can handle. As of now, our group believes that we should implement a queuing system in order to buffer the movement input before InMoov mimics it.

## 3.3 InMoov Robot Arms

The designed of the robot arm is designed by InMoov. The arm is 3D printed and assembled by the group. The arm consists of mechanical parts and servo motors. The servo motors are driven the the micro- controller to help the arm mimic human movements.

The main challenge of the component is that the Leap Motion controller only tracks hand and finger movement. In order to incorporate arm movement, we must improvise. One idea is to only allow 1D arm movement, that is, only allow the arm to rotate along one axis. We can then use one previously untracked measurement (for example, the height of the hand) in order to dictate the movement of the entire arm. This is an overly simplified solution to a very complex problem due to hardware limitations, must it could be a first step. If we wanted to physically pick up items, we would need to include pressure sensors on the fingers in order to detect that we have actually come into contact with the object. This tells InMoov when to stop applying more pressure since the tracked hand will likely be out of sync with InMoov after InMoov comes into contact with an object.

# Milestone and Timeline

The project is divided into 3 main milestones:

**Milestone 1:** Controlled software implementations

There are 2 pieces of software need to be written: the micro- controller firmware, the computers application. The firmware is used to control the robot arms while the application is used to interface the leap motion with the computers.

**Milestone 2:** Interfacing the leap motion and the robot arm

The robot arm is required to mimic the human movement.

**Milestone 3**: Control-over-IP implementations

The control-over-IP software is written.

Figure 2 is the project timeline.

# List of Special Components and Facilities

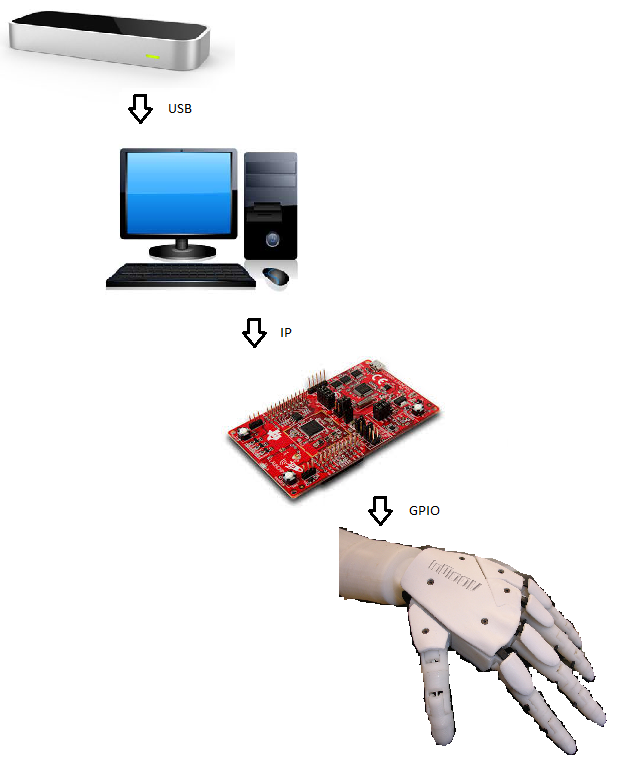
Listed below are the components that we need. These are subject to change at any given point:

* Hardware
  + InMoov open source 3D printed robot (hand, wrist and arm)
  + Leap Motion Controller
  + TI CC3200-LAUNCHXL
  + HS-311 Servo Motors
* Software
  + C++/Python for interfacing with the Leap Motion Controller
  + Berkeley Socket API for communication through IP
  + C on the microcontroller for developing the robotics platform

# Methods

**System Architecture**

The figure below depicts a high level overview of our system:



**Leap Motion Controller**

The Leap Motion controller is a small USB peripheral device that provides accurate hand and finger tracking through the use of the sensor embedded in the device. The Leap provides libraries to query the device for several programming languages including C++, Python, Objective-C, and Javascript. For the purpose of our project, we will likely use C++ to interface our Computer with the Leap device. However, we would like to keep our options open, so we are not discounting Python.

Here are some problems that we have identified that must be addressed at this stage of the system:

* Identifying the relevant data that the Computer needs from the Leap Motion Controller. The Leap Motion provides tracking for the hand and each individual joint of the fingers, vectors representing the normals and direction in which they are facing, and the velocity in which these they move or rotate. Tracking all of this is a bit overkill for our application due to limitations in the movement of InMoov.

**Computer**

The Computer will communicate with the microcontroller through IP using the berkeley socket API. The plan is to process and serialize the relevant data received from the Leap Motion controller, then stream it across the network. The decision of whether to use TCP or UDP for this has not yet been finalized. We would like to avoid using UDP if possible, to save ourselves the trouble of implementing our own reliability and network congestion protocol. Knowing this, we will initially develop our system under the assumption that we will be using TCP, and only switch to UDP if the transmission rate does not meet our expectations.

Here are some problems that we have identified that must be addressed in the communication across IP:

* The IP address of the microcontroller would have to be static and port forwarded on the router in order for an application to communicate with it from a different network.
* One group member who used the CC3200 WiFi shield (the wifi module on our microcontroller) for a different project last year and mentioned that the shield does not work with enterprise WiFi.

To work around both of these limitations, we are planning on bringing our own router (set to bridging mode) to the poster fair demonstration.

**Microcontroller**

Once the relevant data is received on the microprocessor, the data must be processed once again. The microprocessor should calculate the speed at which the joints are moving, and will configure the PWM accordingly in order to simulate that movement on the servo motors.

Here are some problems that must be addressed at this stage of the system:

* Since the InMoov is driven by servo motors, the speed at which it can move will likely be much slower than the potential movement of our hand. We must handle the cases where the tracked hand moves faster than the servos can handle. As of now, our group believes that we should implement a queuing system in order to buffer the movement input before InMoov mimics it.

**InMoov**

If everything goes according to plan, InMoov will successfully mimic the tracked hand. And we will have achieved our main milestones. If time permits, we would also like to incorporate arm movement into our system as mentioned in our extended milestones. Arm movement would allow InMoov to physically pick up items instead of just mimic hand motion.

Here are some issues that must be answered for this to work:

* The Leap Motion controller only tracks hand and finger movement. In order to incorporate arm movement, we must improvise. One idea is to only allow 1D arm movement, that is, only allow the arm to rotate along one axis. We can then use one previously untracked measurement (for example, the height of the hand) in order to dictate the movement of the entire arm. This is an overly simplified solution to a very complex problem due to hardware limitations, must it could be a first step.
* If we wanted to physically pick up items, we would need to include pressure sensors on the fingers in order to detect that we have actually come into contact with the object. This tells InMoov when to stop applying more pressure since the tracked hand will likely be out of sync with InMoov after InMoov comes into contact with an object.