1 Materials and Methods

1.1 Materials

PIC32MX795F512L: A 32 bit micro-controller for performing the control algorithms of the robot. The PIC32MX795F512L comes from SparkFun electronics as part of the UBW32 board, which comes preloaded with a bootloader for programming the board. The PIC32MX795F512L is programmed in C/C++. To make the device easier to use, we loaded an avrdude bootloader onto the device to use the MPIDE software. The MPIDE framework allows us to use Arduino libraries on the PIC32MX795F512L which helps to accelerate development of basic I/O software for the board[5].

PICKIT3: A hardware programmer for the PIC32MX795F512L microprocessor. The pickit3 and program and debug the PIC32 micro-controller. Used to download code onto the UBW32. Useful for installing new bootloaders onto the UBW32 and for restoring broken firmware[6].

MPU6050: A gyro and accelerometer for obtaining information about the kinematics of the arm and the leg. Uses I2C connection to the PIC32MX795F512L for communication[4].

HSR-5498SG: Servos from Hitec. Utilized to control arm and leg joints. The servos require 6-7 Volts. Each servo requires approximately 200mA. The actuation of the arm is accomplished by mini-motors[8].

SSC-32: Servo controller from Lynx motion. The on-board controller is an Atmega168-20PU. The SSC-32 is controlled using serial signals from another microprocessor or the PC. The control signal is a simple byte stream. The controller can specify pin number, servo position, rotation speed, and rotation time. It can power 32 servos simultaneously[2].

1.2 Methods

1.2.1 Preliminary Setup

To program the PIC32MX795F512L, we used the Multi-Platform Integrated Development Environment(MPIDE) environment from chipKIT. To use MPIDE with the program, we had to program the PIC32 with the avrdude bootloader found here: `https://github.com/chipKIT32/PIC32-avrdude-bootloader`. Using the pickit3, we loaded the new bootloader onto the UBW32 and was able to use MPIDE to program the board.

While the other team members were constructing the mechanical components of the legs and arms, I proceeded to set up the basic components of the controller. We set up the MPU6050 using the PIC32 I2C libraries in MPIDE and proceeded to do basic calibration and testing. Then we set up the servos using the SSC-32 controller and the PIC32. Using the PIC32's Universal Asynchronous Receiver/Transmitter(UART) we set up a serial communication channel between the PIC32 and the SSC-32.

1.2.2 Servo Setup

To control the servos we would send a string through the PIC32 UART channel to the SSC-32. The string encoded the pin number, desired position and desired rotation time for one of the 32 servo channels on the SSC-32[8]. There were six servos on each leg of the robot and four on each arm. On the legs, there were two servos per ankle, one on the knee, and three on the hip. In total that gives us six degrees of freedom. The arms were assembled similarly, with one servo on the elbow and three in the shoulder. The servos were powered through the SSC-32 board. Each servo requires around 200mA of current and 6-7V of voltage[2]. We used a 6V power supply to power the servos channels on the SSC-32, and a 9V battery to power the logic circuit of the servos. The PIC32 was powered through a USB connection to the PC.

1.2.3 Walking Algorithm Implementation

The walking algorithms were designed by Zhu Ziqi, a fellow team member. He designed and simulated the algorithms in MATALB and SIMULINK and I implemented the algorithms in C++ on the PIC32MX795F512L. The algorithms are based on Central Pattern Generators, a part of biological neural networks which generate rhythmic control signal for human motion[1]. The servos are controlled using sine waves of varying shapes to generate patterns for walking. Ziqi performed simulations to to select the best parameters for the sine waves. For our preliminary walking algorithms we only considered the knee and hip joints, keeping the other joints rigid. The SSC-32 position control signal requires a integer ranged from 500 to 2500, corresponding to 180° of motion[8]. Due to the mechanical construction of the arm and leg joints, which mimicked human structure, most joint servos were limited to a range between 1250 and 2500. The output from the sine waves would be converted to a range in the SSC-32's output and then sent to the servos.

$$sine_{knee\_joint} = sin(eqn\_pending) \tag{1}$$

$$sine_{hip\_joint} = sin(eqn\_pending) \tag{2}$$

Equations 1 and 2 show the equations for calculating the position of each servo as a function of time. The positions of the legs are calculated by the PIC32 and sent to the SSC-32.

1.2.4 Leg Balance Algorithm

Waiting for team member to finish simulations of control algorithms. Likely will use the MPU6050 to get the information about balance.

1.2.5 Arm Control Algorithm

The arms are controlled using a Fuzzy Inference System. A fuzzy inference system comprises of a fuzzifier, inference system, and defuzzifier. The fuzzyfier takes inputs and out converts them to fuzzy values of a set of fuzzy variables for the inference system. In this case the input is the desired position of the arm in 3D space. The inference system is a set of if else statements that determine a desired output from the fuzzy values. The defuzzifier then converts this output to a real output to our servos[7]. To design this system we will take two approaches, one through simulation and one through machine learning. I will work on the Machine Learning approach. MATLAB has neural network algorithms that will generate fuzzy inference systems. To do this we will generate a dataset based on the forward kinematics of the robot arm, by giving it the angles of the arm joints and calculating the arm position as a result of the joint angles. Inputing this dataset into a machine learning framework in MATLAB, we can generate a simple fuzzy inference system to control the robotic arm[3]. To further refine the control algorithm, we will need to hand tune and simulate the fuzzy inference system, as well as utilize gyro, accelerometer, and IR sensors to help guide the arm and avoid obstacles.

1.2.6 Computer Vision

To be completed... uses serial communication from APU to PIC32 to transmit desired arm coordinates.

1.2.7 Control of Hand

Not sure if implementation of hand control is needed for project.

References

[1] Scott L Hopper. *Central Pattern Generator*. Ohio University. Feb. 2000. URL: `http://crab-lab.zool.ohiou.edu/hooper/cpg.pdf`.

[2] Jun Hee Lee. *General Specifications of HSR-5498SG Digital Robot Servo*. Hitec. June 2006. URL: http://www.robotshop.com/media/files/pdf/hitec-hsr-5498sg-digital-servo-specsheet.pdf.

[3] *Modeling Inverse Kinematics in a Robotic Arm*. MathWorks. 2014. URL: http://www.mathworks.com/help/fuzzy/examples/modeling-inverse-kinematics-in-a-robotic-arm.html.

[4] *MPU6050 Datasheet*. InvenSense Inc. Aug. 2013. URL: http://invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf.

[5] *PIC32MX5XX/6XX/7XX Datasheet*. Microchip. Nov. 2012. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf.

[6] *PICkit 3 Programmer/Debugger Users Guide*. Microchip. Jan. 2008. URL: https://www.sparkfun.com/datasheets/Programmers/PICkit_3_User_Guide_51795A.pdf.

[7] Farzin Piltan et al. "Design Adaptive and Fuzzy Inference and Sliding Mode and Algorithm: Applied and to and Robot Arm". In: *International Journal of Robotics and Automation* 2.5 (2011), pp. 283–297.

[8] *User Manual SSC-32 Ver 2.0*. Lynxmotion, Inc. 2005. URL: http://www.swarthmore.edu/NatSci/ceverba1/Class/e5/E5Lab2/ssc-32%20lynxmotion%20manual.pdf.