This project was created primarily as a self-study in the field of robotics. Its intent was to incorporate the many strands that have been taught through the course of schooling. To an extent it has accomplished just that; the project is a merger of electronics, motion control, mechanical design and programming. Although there is still much to expand upon and incorporate into the project, this has served as a great starting point.

Summary

This project involved the creation of a quadruped robot. The robot parts were designed in AutoCAD 2D, extruded into 3D models and printed at the college. The project was a tool for me to learn more about mechanical design, creating something from scratch that is capable of moving, electrical design and programming. The project incorporates two different programming languages, C# (Arduino IDE) and Python. It has also increased my knowledge in computer science as I have been teaching myself more and more about Linux through the use of the raspberry pi. The robot itself has 3 main modules for the time being

- Arduino Mega R3 2560
- Raspberry Pi 2
- 16 Channel 12-Bit PWM/Servo driver

The project has presented many challenges, but each one presents a unique learning opportunity and another tool added to my troubleshooting skills.

Obstacles encountered

- Analog feedback was giving ambiguous readings, problem was due to lack of ground between servo driver and Arduino
- Edges of servo body collided with servo brackets, servo body was filed down
- Female to male jumpers that were used to connect the servos to the servo driver were too stiff, servo extension cables were used in its place
- The epoxy that was used to join the servo brackets failed in one instance, the servo bracket was rejoined using epoxy and has yet to fail again. If this becomes problematic in the future, screws will be used along with the epoxy to fasten the two brackets in place.
- Using the PWM library from ad fruit proved unintuitive, a function was created in Arduino to map the PWM library to degrees
- The servo driver's voltage was dropping while all servos were active. A 470 μF Capacitor was soldered between the + and – supply of the board.

Build procedure

-First conceptual drafts – 4 Hrs.
-Drafts put into auto cad – 1Hrs.
-AutoCAD 2d drafts perfected – 10+ Hrs.
- 3D models of auto cad parts producers, exported into STL files – 20+ Hrs.
- 3D parts are printed
- Motors are sized controllers picked – 2 hrs.
- Servo brackets are glued together – 2 hrs.
-Servo horns are mounted into parts – 1 hrs.
-Body holes bored out to accommodate bearings, bearings installed – 1 hr.
-Sides of body filed down to stop servo collisions – 2 hrs.
-Servos installed into legs, femur brackets – 1 hr.
-All servos are centered using a simple Arduino program – 1hrs.
-Servos mounted into their respective horns while centered – 4 hrs.
-Leg assemblies are mounted into the bottom portion of the body – 1hr.
-Servo driver soldered together, installed in the bottom portion of the body – 1hr.
-Body standoffs installed – 1hr.
- Top portion of body mounted to bottom portion. – 3hrs.
- Arduino and raspberry pi are mounted to the top of the robot – 2hrs.
- Wiring – 10hrs.
- Programming- 60+ Hours
- Raspberry pi implementation – 50+ Hours
Total time 177+ Hours

Code and algorithms

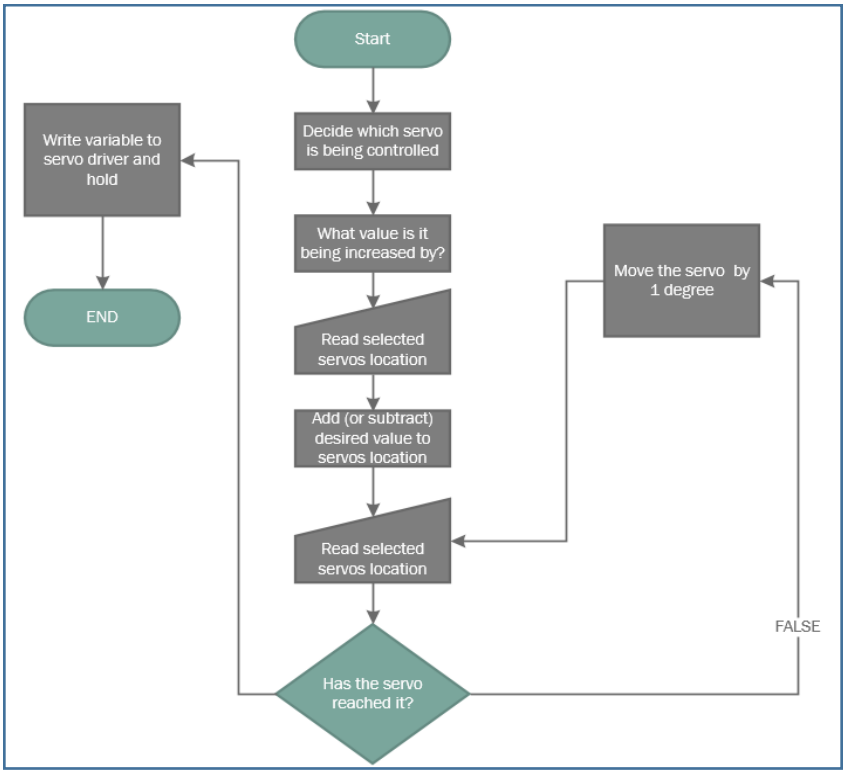
The Arduino is loaded with a sketch that has 3 different walking algorithms (Forward, Left turn, Right turn). It also has several functions to make the coding of the walking algorithm easier.

Pulses to Degrees: DTP

This function maps the max and min amount of pulses sent to servo via the 12 bit PWM driver and converts it to an angular value which is easier to understand

Positive angular increase, Negative angular increase: P_INC, N_INC

This function takes in two values; the servo you desire to move, and the amount you want to increment it by. The function reads the current positions of the servo motor using the feedback function. Then it adds or subtracts the value you want to move it, then it moves itself until it has reached it, constantly checking the feedback. The flow chart for this can be seen on the right.



There a few more functions but they all perform essentially the same routine the Arduino code is posted here.

```
#include <Wire.h>   // include the i2c library
#include <Adafruit_PWMServoDriver.h>    // include the ad fruit servo driver library

/* set adrees for i2c comm(servosheild) to 0x40 jumper must be installed on board to modify */
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();


/************************************************************
Set up pins for servo feedback and servo signal
CREATE FeedbackArray
B = BODY, F = FEMUR , L = LEG
************************************************************/
int FeedbackArray[12];
// BLUE GROUP
int BB_F = A0;
int BF_F = A1;
int BL_F = A2;
// RED GROUP
int RB_F = A3;
int RF_F = A4;
int RL_F = A5;
// GREEN GROUP
int GB_F = A6;
int GF_F = A7;
int GL_F = A8;
//YELLOW GROUP
int YB_F = A9;
int YF_F = A10;
int YL_F = A11;

/*********************************************************
FUNCTION TO CALIBRATE FEEDBACK FROM SERVO MOTORS
CALIBRATED BETWEEN 25 - 165
*********************************************************/
int calibrate(){
for (int p = 0 ; p < 12; p++){
/* Sets all servos to 90 each time the next servo is calibrated */
pwm.setPWM(BB_S, 0, DTP(90));
pwm.setPWM(BF_S, 0, DTP(90));
pwm.setPWM(BL_S, 0, DTP(90));
pwm.setPWM(GB_S, 0, DTP(90));
pwm.setPWM(GF_S, 0, DTP(90));
pwm.setPWM(GL_S, 0, DTP(90));
pwm.setPWM(RB_S, 0, DTP(90));
pwm.setPWM(RF_S, 0, DTP(90));
pwm.setPWM(RL_S, 0, DTP(90));
pwm.setPWM(YB_S, 0, DTP(90));
pwm.setPWM(YF_S, 0, DTP(90));
```

```
                                        pwm.setPWM(YL_S, 0, DTP(90));
for (uint16_t i = 25; i < 165; i++){pwm.setPWM(p, 0, DTP(i));}      // START SERVO AT 25 DEG MOVE TO
                                                                    165
    delay(3000);                                                    // DELAY TO ENSURE IT HAS REACHED IT
                                                Serial.print(" i am ..");
    Serial.print(p);                                                // PRINT OUT WHAT SERVO NUM ITS MOVING
                                                Serial.print("...");
                                               int Y = analogRead(p);
                        Serial.println(Y);                                  // PRINT OUT RAW DATA
                                                delay(3000);
 for ( uint16_t i = 165; i > 25; i--){ pwm.setPWM(p, 0, DTP(i));}    // START SERVO AT 165 DEG MOVE
                                                                    TO 25
     delay(3000);                                                   // DELAY TO ENSURE IT HAS REACHED IT
                                               Serial.print(" i am ..");
     Serial.print(p);                                               // PRINT OUT WHAT SERVO NUM ITS MOVING
                                                Serial.print("...");
                                              int X = analogRead(p);
                    Serial.println(X);                                  //PRINT OUT VALUE OF SERVO
                                               delay(3000);}}

                    /************************************************************************
 THIS SECTION DEFINES PULSE LENGTH COUNT FOR SERVO @ MAPS THEM TO DEGREES
                    ************************************************************************/

                    #define SERVOMIN  130 // this is the 'minimum' pulse length count (out of 4096)
                    #define SERVOMAX  620 // this is the 'maximum' pulse length count (out of 4096)

                    /*!!!!!!!!! CONVERTS DEGREES INTO PULSE'S!!!!!!!!!!!!!!!!!!*/
                                        int DTP(int Deg){
                                            uint16_t Result;
                                    Result = map(Deg,0, 180, SERVOMIN,SERVOMAX);
                                            return Result;}

                    /*****************************************************************
                    FUNCTION TO GATHER FEEDBACK INFORMATION FROM SERVO
                    *****************************************************************/
                                        int Feedback() {
                                    FeedbackArray[0] = analogRead (BB_F);
  FeedbackArray[0] = map(FeedbackArray[0], 175,475,25,165);  // THIS MAPS THE VALUES TAKEN
 FeedbackArray[1] = analogRead (BF_F);                              // FROM THE CALIBRATION ROUTINE
FeedbackArray[1] = map(FeedbackArray[1], 176,476,25,165); // TO DEGREES. READS THE SERVOS
   FeedbackArray[2] = analogRead (BL_F);                            //VALUE AND STORES THAT IN AN
    FeedbackArray[2] = map(FeedbackArray[2], 171,468,25,165); // ARRAY TO BE READ BY THE
   FeedbackArray[3] = analogRead (RB_F);                            // THE MOVEMENT FUNCTIONS
                        FeedbackArray[3] = map(FeedbackArray[3], 176,483,25,165);
                                    FeedbackArray[4] = analogRead (RF_F);
                        FeedbackArray[4] = map(FeedbackArray[4], 173,471,25,165);
                                    FeedbackArray[5] = analogRead (RL_F);
                        FeedbackArray[5] = map(FeedbackArray[5], 179,484,25,165);
                                    FeedbackArray[6] = analogRead (GB_F);
                        FeedbackArray[6] = map(FeedbackArray[6], 191,480,25,165);
```

```
FeedbackArray[7] = analogRead (GF_F);
FeedbackArray[7] = map(FeedbackArray[7], 178,483,25,165);
FeedbackArray[8] = analogRead(GL_F);
FeedbackArray[8] = map(FeedbackArray[8], 183,506,25,165);
FeedbackArray[9] = analogRead(YB_F);
FeedbackArray[9] = map(FeedbackArray[9], 176,476,25,165);
FeedbackArray[10] = analogRead(YF_F);
FeedbackArray[10] = map(FeedbackArray[10], 179,488,25,165);
FeedbackArray[11] = analogRead(YL_F);
FeedbackArray[11] = map(FeedbackArray[11], 176,480,25,165);
}
/*****************************************************************************
!!!!!!!!!!!!!!FUNCTION TO PRINT OUT FEEDBACK TO THE SERIAL MONITOR!!!!!!!!!!!!!!
*****************************************************************************/
int printfeedback() {
for (int x = 0 ; x < 12 ; x++){            // MOVE THROUGH FEEDBACK ARRAY
Serial.print(F("i am slot no..."));        // WHAT SLOT AM I LOOKING AT
Serial.print(x);
Serial.print(F("....my value is....."));   // WHAT VALUE IS IT
Serial.println(FeedbackArray[x]);
}}
/*****************************************************************************
!!!!!!!!!!!!!!!!!!!STARTING POISTION OF ROBOT!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*****************************************************************************/
int STARTPOS(){
pwm.setPWM(BB_S, 0, DTP(65));
pwm.setPWM(BF_S, 0, DTP(130));     // VALUES OF THIS ARE ARBITARRY
pwm.setPWM(BL_S, 0, DTP(120));     // DETERMINED BY VISUAL INSPECTION
pwm.setPWM(GB_S, 0, DTP(95));
pwm.setPWM(GF_S, 0, DTP(130));
pwm.setPWM(GL_S, 0, DTP(110));
pwm.setPWM(RB_S, 0, DTP(120));
pwm.setPWM(RF_S, 0, DTP(50));
pwm.setPWM(RL_S, 0, DTP(50));
pwm.setPWM(YB_S, 0, DTP(90));
pwm.setPWM(YF_S, 0, DTP(50));
pwm.setPWM(YL_S, 0, DTP(50));
delay(1000);}
/*****************************************************************************
FUNCTION TO MOVE THE ROBOT TO A DESIRED POSTION
*****************************************************************************/
int MOVE(char SERVO, int DESIRED_POS){
uint8_t servoNum;
if (SERVO == BB_S){servoNum = 0;}
if (SERVO == BF_S){servoNum = 1;}  // MAP THE SERVO INPUT CHAR TO A NUMBER
if (SERVO == BL_S){servoNum = 2;}
if (SERVO == RB_S){servoNum = 3;}
if (SERVO == RF_S){servoNum = 4;}
if (SERVO == RL_S){servoNum = 5;}
if (SERVO == GB_S){servoNum = 6;}
if (SERVO == GF_S){servoNum = 7;}
```

```
if (SERVO == GL_S){servoNum = 8;}
if (SERVO == YB_S){servoNum = 9;}
if (SERVO == YF_S){servoNum = 10;}
if (SERVO == YL_S){servoNum = 11;}
    Feedback();                                          // READ THE POSITIONS OF SERVOS
int CurrentPos = FeedbackArray[servoNum];      // CREATE A VARIABLE, SET IT TO POS OF SERVO

if( CurrentPos < DESIRED_POS){
for (uint16_t MOVER = CurrentPos; MOVER < DESIRED_POS; MOVER+=5)
{pwm.setPWM(servoNum, 0, DTP(MOVER));
delay(5);}
}

if( CurrentPos > DESIRED_POS){
for (uint16_t MOVER = CurrentPos; MOVER > DESIRED_POS; MOVER-=5)
{pwm.setPWM(servoNum, 0, DTP(MOVER));
delay(15);}
}
}
/*****************************************************************************
FUNCTION TO INCREMENT A SERVO BY WHATEVER VALUE IS PUT IN (POSITIVE ROTATION)
*****************************************************************************/
int POS_INC(char SERVO, int INC_VAL){                    // FUNCTION TAKES IN SERVO
CHARACTER, AND VALUE TO INCREMENT BY
uint8_t servoNum;
if (SERVO == BB_S){servoNum = 0;}
if (SERVO == BF_S){servoNum = 1;}
if (SERVO == BL_S){servoNum = 2;}
if (SERVO == RB_S){servoNum = 3;}
if (SERVO == RF_S){servoNum = 4;}
if (SERVO == RL_S){servoNum = 5;}
if (SERVO == GB_S){servoNum = 6;}
if (SERVO == GF_S){servoNum = 7;}
if (SERVO == GL_S){servoNum = 8;}
if (SERVO == YB_S){servoNum = 9;}
if (SERVO == YF_S){servoNum = 10;}
if (SERVO == YL_S){servoNum = 11;}

Feedback();
int CurrentPos = FeedbackArray[servoNum];
int GOTO = INC_VAL + CurrentPos;

for (uint16_t MOVER = CurrentPos; MOVER < GOTO; MOVER +=1)
{
pwm.setPWM(servoNum, 0, DTP(MOVER));

}
}
/*****************************************************************************
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!END OF FUNCTION!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*****************************************************************************/
```

```
/*****************************************************************************
FUNCTION TO INCREMENT A SERVO  BY WHATEVER VALUE IS PUT IN (NEGATIVE ROTATION)
*****************************************************************************/
int NEG_INC(char SERVO, int INC_VAL){                    // FUNCTION TAKES IN SERVO CHARACTER, AND VALUE TO INCREMENT BY
uint8_t servoNum;                                        // THIS VALUE IS FOR WHICH SERVO IS TO BE MOVED
if (SERVO == BB_S){servoNum = 0;}                        // THIS ASSIGNES SERVO NUMBERS TO WHATEVER IS PUT IN
if (SERVO == BF_S){servoNum = 1;}
if (SERVO == BL_S){servoNum = 2;}
if (SERVO == RB_S){servoNum = 3;}
if (SERVO == RF_S){servoNum = 4;}
if (SERVO == RL_S){servoNum = 5;}
if (SERVO == GB_S){servoNum = 6;}
if (SERVO == GF_S){servoNum = 7;}
if (SERVO == GL_S){servoNum = 8;}
if (SERVO == YB_S){servoNum = 9;}
if (SERVO == YF_S){servoNum = 10;}
if (SERVO == YL_S){servoNum = 11;}

Feedback();                                              // READ THE CURRENT POSITION OF ALL MOTORS
int CurrentPos = FeedbackArray[servoNum];                // ACCSES ARRAY FOR MOTOR IN QUESTION
int GOTO = CurrentPos - INC_VAL  ;                       // SUBTRACT THE VALUE

for (uint16_t MOVER = CurrentPos; MOVER > GOTO; MOVER -=1)          // MOVE TO THE COMPUTED VALUE
{
pwm.setPWM(servoNum, 0, DTP(MOVER));

}
}

/*****************************************************************************
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!END OF FUNCTION!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*****************************************************************************/
int walk_seq1 (){


/// SEQUENCE ONE
POS_INC(RF_S, 5); // PUSH RED FEMUR TOWARS GROUND
POS_INC(YF_S, 5); // PUSH YELLOW FEMUR TOWARDS GROUND

// NEG_INC(RL_S, 10);  // SLIDE RED LEG TOWARDS BODY
// NEG_INC(YL_S, 10);  // SLIDE YELLOW LEG TOWARDS BODY

NEG_INC(GF_S, 5);  // PUSH GREEN FEMUR TOWARDS GROUND
```

```
POS_INC(BF_S, 5);  // PULL BLUE FEMUR UP

delay(1000);

POS_INC(GF_S, 50);  // LIFT GREEN FEMUR OFF GROUND

delay(1000);
POS_INC(GB_S, 60);   // MOVE GREEN LEG ASEMBELY TOWARDS FRONT
delay(1000);
NEG_INC(GF_S, 30);   // BRING GREEN FEMUR BACK TO GROUND
delay(1000);
NEG_INC(RF_S, 5);   // BRING RED FEMUR BACK UP
NEG_INC(YF_S, 5);  // BRING YELLOW FEMUR BACK UP
//POS_INC(RL_S, 10);  // MOVE RED LEG AWAY FROM BODY
//POS_INC(YL_S, 10);    // MOVE RED LEG AWAY FROM BODY

NEG_INC(BF_S, 5);    // BRING BLUE FEMUR DOWN

delay(1000);
NEG_INC(YF_S,30);
delay(100);
NEG_INC(YB_S,50);  // MOVE YELLOW LEG ASSEMBLY TO FRONT
POS_INC(YF_S,30);   // MOVE YELLOW FEMUR DOWN
delay(1000);

// THRUST FORWARD

NEG_INC(GB_S,50);
POS_INC(YB_S,50);

POS_INC(BB_S,50);
NEG_INC(RB_S,50);
delay(1000);

POS_INC(RF_S,5);  // BRING RED FEMUR DOWN
POS_INC(YF_S,5); // BRING YELLO FEMUR DOWN

delay(1000);


POS_INC(BF_S,50);  // BRING BLUE FEMUR UP
NEG_INC(BB_S,50);  // MOVE BLUE BODY
delay(100);
NEG_INC(BF_S,30);
delay(100);


NEG_INC(RF_S,5);  // BRING RED FEMUR up
NEG_INC(YF_S,5); // BRING RED FEMUR up
delay(1000);
```

```
NEG_INC(BF_S,10);
NEG_INC(BF_S,10);
delay(100);

NEG_INC(RF_S,30);
POS_INC(RB_S,50);
POS_INC(RF_S,30);
delay(199);
STARTPOS();
delay(1000);
}


void setup() {

pwm.begin();
pwm.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
Serial.begin(115200);
STARTPOS();
delay(9000);


}

void loop() {
walk_seq1();
}
```