Group: 20
Stanley Wang
Steven Salah-Eddine
Andrew Boutros

Design of a Bio-Inspired Monopedal Jumping Robot

ME 102B – Mechatronics Design, Fall 2021
Due: 12/12/21

Opportunity

In the past years, rapid advances have been made in the field of developing agile robotics systems. Notable examples include SALTO – a monopedal jumping robot from the UC Berkeley Biomimetic Millisystems Lab – and the "Cheetah" – a backflipping quadruped developed by MIT. These robotics systems have the potential to be vastly important in numerous applications, ranging from critical high-risk roles such as search-and-rescue to assisting in everyday tasks. In this project, we were inspired by the ongoing research on the biomechanics of squirrel locomotion. We present the development of a preliminary mechatronics system derived from the kinematics of SALTO. This system will provide a baseline to enable the future development of a novel squirrel-inspired jumping robot.

System Functionality

Our mechatronics system will serve as a preliminary prototype to demonstrate the kinematics of a jumping-leg linkage system. The actuation of the jumping leg will be controlled through a user interface consisting of a button to arm the robot, a button to initiate the jump, and a potentiometer to adjust the intensity of the jump. Furthermore, an accelerometer will be integrated as a safety feature, in order to ensure that the robot will only jump while it is in a properly-oriented upright position.

We set an ambitious initial objective of having a robotic system capable of jumping over 1 meter high, with the ability to balance itself on takeoff and landing. However, the complexity of utilizing a high-performance actuator as well as a control scheme for balancing the robot was beyond our development capabilities due to time and budgetary constraints. Nevertheless, we successfully achieved a jumping leg mechanism capable of jumping up to 15 cm attached to a vertical guide rail for stabilization.

Physical Prototype

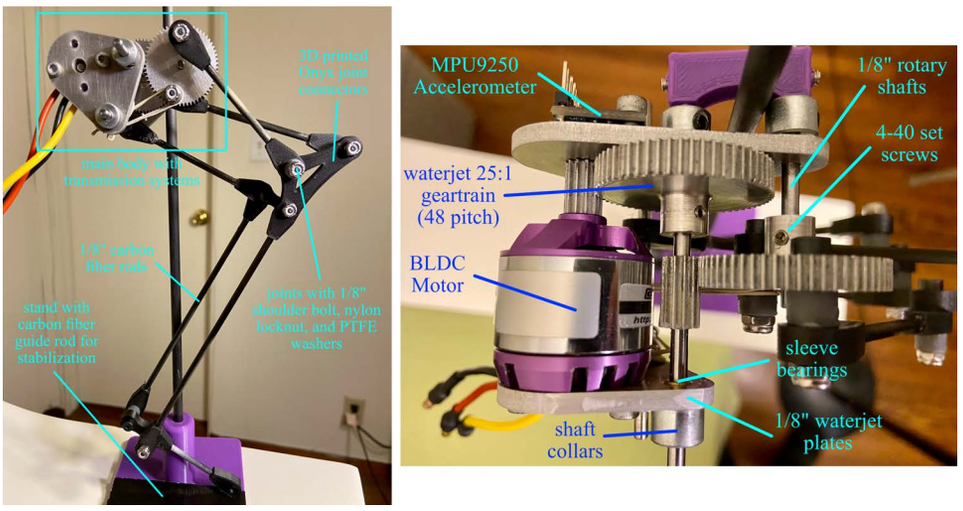


Figure 1: Key components of physical prototype assembly

Function Critical Decisions

The selection of appropriate components for our project was highly critical, since we required optimized high-performance systems in the mechanical design. For our linkage, we initially prototyped out of cast acrylic, which we found to be far too brittle and heavy for our application. Instead, the final linkage system is constructed out of solid carbon fiber rods and Onyx (carbon fiber reinforced Nylon) for the best possible rigidity and strength-to-weight ratio. Instead of integrating large bearings at each joint of our linkage, precision-toleranced shoulder bolts combined with PTFE washers was found to adequate at producing low-friction, wiggle-free joints.

For the body of the robot containing the power transmission systems, we needed a highly weight-optimized package capable of containing both our BLDC motor and a complete 25:1 geartrain. Taking inspiration from A3, we constructed the body to consist of waterjet aluminum plates, with both the motor and shafts of the geartrain effectively functioning as the standoffs. Due to the geometry of the BLDC motor (where the entire outside shell of the motor spins with the exception of a small base), it was difficult to incorporate a locating feature when mounting the motor. Instead, M3 bolts were used to secure the motor within acceptable tolerances. For the geartrain, 48 pitch gears were waterjet and carefully machined with both the lathe and mill. While these gears are fairly heavy, they proved to be the best option in terms of strength (3D-printed gears were too weak). The gears are affixed to steel rotary shafts with 4-40 set screws, and sleeve bearings with shaft collars are used to properly constrain the shafts.

In order to assess the survivability of our geartrain , we can determine the torque and forces experienced by the components. We use a 1300kV BLDC motor with a maximum power output of 275W, connected to a 12V, 3A power supply. Thus, the nominal speed of the motor (assuming it is not under substantial load or stalling) is:

$$\omega = 1300 \text{ kV} \times 12 \text{ V} = 15600 \text{ RPM}$$

The maximum power of the motor is limited by our power supply $P_{max} = 12 \text{ V} \times 3 \text{ A} = 36 \text{ W}$. Thus, energy balance of the motor at constant speed yields a maximum sustainable torque of:

$$\tau_{in} = \frac{P_{max}}{\omega} = \frac{36 \text{ W}}{1633.63 \text{ rad/s}} = 0.022 \text{ N} \cdot \text{m}$$

Our gear teeth have a pressure angle (θ) is 14.5^o, and the radius of the input pinion is r_1 = 0.25 inches. Thus, the torque balance on the input pinion is seen below:

$$\vec{\tau_{in}} + \vec{r_1} \times \vec{F_1} = \vec{0}$$

$\therefore$

$$F_1 = \frac{\tau_{in}}{r_1 \cos\theta} = \frac{0.022 \text{ N} \cdot \text{m}}{0.00635 \text{ m } \cos(14.5^o)} = 3.58 \text{ N}$$

The output gear radius is r_2= 1.25 inches. The torque balance on the output gear is similar to the torque balance on the input pinion and seen below:

$$\vec{\tau_{out}} + \vec{r_3} \times \vec{F_3} = \vec{0}$$

where $\tau_{out} = 25 \times \tau_{in} = 0.55N.m$ since our gear ratio is 25:1. Thus,

$$F_3 = \frac{\tau_{out}}{r_3 \cos\theta} = \frac{0.55 \text{ N} \cdot \text{ m}}{0.0317 \text{ m } \cos(14.5^o)} = 17.89 \text{ N}$$

Therefore, the vertical component of the total reaction force on the compound gear $F_{2y} = F_1 \cos(\theta) + F_3 \cos(\theta) = 3.58 \cos(\theta) \text{ N} + 17.89 \cos(\theta) \text{ N} = 20.79 \text{ N}$, and the horizontal components is $F_{2x} = F_1 \sin(\theta) + F_3 \sin(\theta) = 3.58 \sin(\theta) \text{ N} + 17.89 \sin(\theta) \text{ N} = 5.38 \text{ N}$, so the total force $F_2 = \sqrt{20.79^2 + 5.38^2} = 21.47 \text{ N}$. For an 1/8" shaft, this corresponds to an average shear stress of 2.712 MPa, which is substantially below the yield strength of steel!

Circuit Diagram

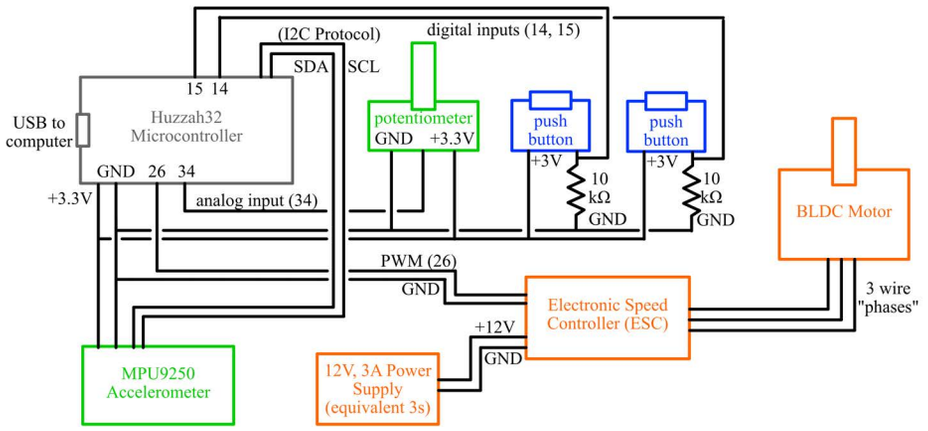


Figure 2: Circuit diagram of key electronic components in control circuitry

State Transition Diagram

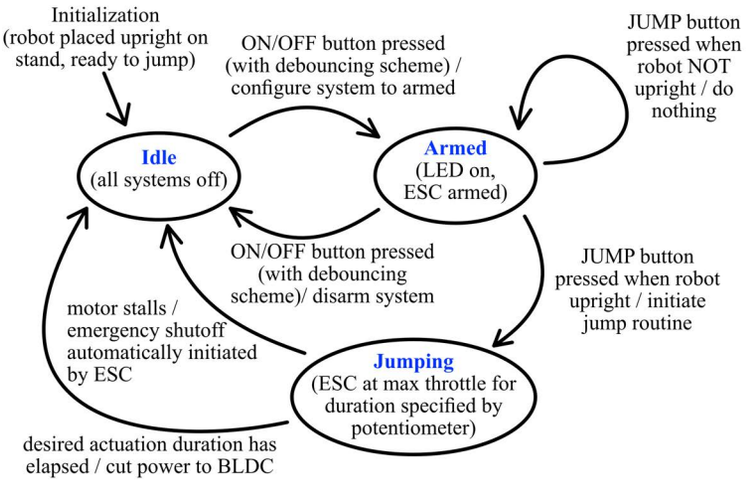


Figure 3: State transition diagram illustrating characteristic operation of system

Reflection

We started out this project with extremely high expectations and ambitious performance goals. Unfortunately, we quickly realized that our initial vision would not be achievable with our limited time and resources, and thus adjusted our design goals accordingly while maintaining compliance with key project requirements. Although we encountered many obstacles in the realization of our system, we were nevertheless able to assemble a system exceeding expectations and capable of performing small vertical jumps. A substantial effort was invested into realizing the physical prototype of our design, and many hours were spent in the machine shop as we created our linkage system and geartrain. In the future, we hope to be able to iterate further on our existing design and take additional inspiration from squirrel biomechanics. Specifically, additional stability throughout the jump and landing process as well as further optimization of existing mechanical systems are some of our top priorities.

Author Contributions

Stanley Wang
CAD design and kinematic animations of finalized mechanical systems. Development of control circuitry and event-driven program in Arduino IDE, Manufacturing and testing of physical prototype. Principal artist of diagrams and illustrations, and primary author of manuscript.

Steven Salah-Eddine
Contributed advice to design concepts. CAD design of preliminary iterations. Parts-acquisition from McMaster Carr. Logistics manager of ASME Maker Grant deliverables and Bill of Materials. Editor of final project demonstration video.

Andrew Boutros
Contributed to manufacturing processes in ME machine shop. Theoretical validation of designs with function-critical calculations. Author of the reflection.

Appendix A: Bill of Materials

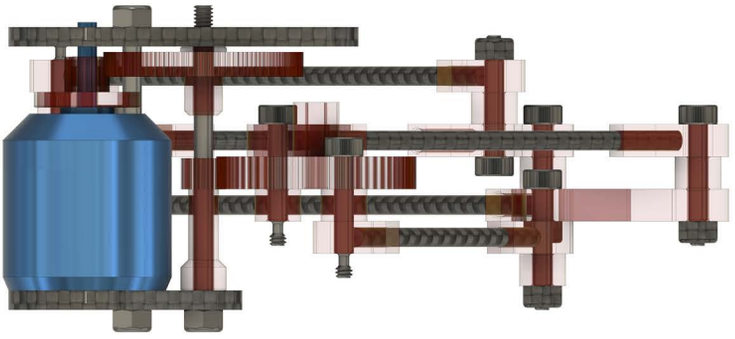
Bill Of Materials

Item Name	Description	Purchase Justification	Serial Number / SKU	Price (ea.)	Quantity	Vendor
Teflon Washers	Chemical-Resistant PTFE Plastic Washer	Used to remove adhesion between joints of our Robots	95630A420	$ 3.78	1	McMaster Carr
Alloy Steel Shoulder Screw	1/8" Shoulder Diameter, 5/8" Shoulder Length, 4-40 Thread	Used to bolt two components together	91259A493	$ 2.00	10	McMaster Carr
18-8 Stainless Steel Nylon-Insert Locknut	18-8 Stainless Steel Nylon-Insert Locknut, 4-40 Thread Size	Locknut used to keep the bolt in place	91831A005	$ 3.61	1	McMaster Carr
Carbon Fiber Rod	1/8" Diameter, 24" Long	The Rod is used for the legs of the robot	2153T16	$ 12.00	1	McMaster Carr
30A Electronic Speed Controller (ESC)	30A RC Brushless Motor Electric Speed Controller	Speed Controller for our BLDC Motor	B071GRSFBD	$ 16.99	1	Amazon
Brushless DC (BLDC) Motor	DYS 1300KV Brushless DC Motor (BLDC)	Motor to drive actuation	B077HLPP4N	$ 16.99	1	Amazon
Jacobs Material Store Clear Acrylic	1/8" x 16" x 32" sheet	Acrylic for prototyping preliminary linkage system	A181632	$ 27.60	1	Jacobs Material Store
Stainless steel screw	18-8 Stainless Steel Flat-Tip Set Screws, 4-40 Thread, 1/16" Long, Packs of 50	To attach gear train on rotary shafts	94355A215	$ 6.59	1	McMaster Carr
Rotary Shaft	Rotary Shaft, 12L14 Carbon Steel, 1/8" Diameter, 3" Long	to constrain gear train	1327K93	$ 3.32	5	McMaster Carr
Stainless steel screw	18-8 Stainless Steel Flat-Tip Set Screws, 4-40 Thread, 1/8" Long, Packs of 50	To attach gear train on rotary shafts	94355A130	$ 6.10	1	McMaster Carr
Shaft Collars	Set Screw Shaft Collar For 1/8" Diameter, Zinc-Plated 1215 Carbon Steel	Secure rotary shafts in place in geartrain	6432K17	$ 1.19	5	McMaster Carr
Rotary Shaft	Rotary Shaft, 12L14 Carbon Steel, 1/8" Diameter, 3" Long	to constrain gear train	1327K93	$ 3.32	3	McMaster Carr

Socket Head Screws	Zinc-Aluminum-Coated Alloy Steel Socket Head Screw, M3 x 0.5 Mm Thread, 6 Mm Long	For mounting motor to the metal plate	91274A102	$ 5.38	1	McMaster Carr
Socket Head Screws	18-8 Stainless Steel Socket Head Screw, M2 x 0.4 Mm Thread, 8 Mm Long	For mounting input pinion gear to the motor	91292A832	$ 5.70	1	McMaster Carr
Elastic Cord	Mil. Spec. Elastic Cord, 1/8" Diameter, 10 Feet Long	Elastic element to act as torsional spring in driving geartrain	2203N11	$ 6.67	1	McMaster Carr
Locknuts	18-8 Stainless Steel Nylon-Insert Locknut, 5-40 Thread Size	Nuts for securing ends of rotary shafts (supplementary/alternative to shaft collar)	91831A006	$ 5.11	1	McMaster Carr
Sleeve bearings	Oil-Embedded Bronze Sleeve Bearing For 1/8" Shaft Diameter And 3/16" Housing Id, 1/8" Long	Bearings to reduce the friction of rotary shafts in geartrain	6391K611	$ 0.67	8	McMaster Carr
Onyx filament joints and connectors	Markforge 3D printed 13 joints and connectors	The joints and connectors were used to connect the linkages together	n/a	$ 7.02	1	UC Berkeley Machine Shop

Total Cost	$181.41
Total Cost(with Shipping & Tax)	$220.21

Appendix B: CAD Models



Appendix C: Event-Driven Program

```
#include "MPU9250.h" // library for working with accelerometer module

 // Analog input for potentiometer
 #define POT 32
 // Digital output for built-in LED
 #define LED_PIN 13
 // Digital input for ON/OFF Button
 #define BTN_1 15
 // Digital input for JUMP Button
 #define BTN_2 14
 // Analog output for ESC PWM signal
 #define PWM_1 26


// Setup MPU9250 (Accelerometer Module)
MPU9250 mpu;
// Accelerometer Calibration Offsets (predetermined such that A_net     0 when upright)
float x_bias = -3.13;
float y_bias = -0.51;
float z_bias = -0.32;


// Setup PWM for Controlling ESC
const int freq = 50;
const int ledChannel_1 = 1;
const int resolution = 12; // 12 bit (2^12 = 4096)
int command = 0;


// Important State Variables
volatile bool ON_OFF_pressed = false; // detect if ON/OFF button pressed
volatile bool JUMP_pressed = false; // detect if JUMP button pressed
volatile bool debouncingtimer = true; // for debouncing buttons (true when button press ok)
volatile bool motortimer = true; // for determining how long to power BLDC (open-loop)
byte state = 0;
 // 0 = idle
 // 1 = armed
 // 2 = jumping


 // Interrupt-Service Routine Function (ISR) for ON/OFF Button
 void IRAM_ATTR isr_1() {
  if (debouncingtimer) {
    ON_OFF_pressed = true;
  }
 }


// Interrupt-Service Routine Function (ISR) for JUMP Button
 void IRAM_ATTR isr_2() {
  if (debouncingtimer) {
    JUMP_pressed = true;
  }
 }


// Timer Interrupt for Debouncing (shared by both buttons)
hw_timer_t * timer0 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  debouncingtimer = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}
void TimerInterruptInit0() {  //The timer simply counts the number of Tic generated by the
    quartz. With a quartz clocked at 80MHz, we will have 80,000,000 Tics.
  portENTER_CRITICAL_ISR(&timerMux0);
  debouncingtimer = false;
  portEXIT_CRITICAL_ISR(&timerMux0);
  timer0 = timerBegin(0, 80, true); // divides the frequency by the prescaler: 80,000,000 /
    80 = 1,000,000 tics / sec
  timerAttachInterrupt(timer0, &onTime0, true);    // sets which function do you want to
    call when the interrupt is triggered
  timerAlarmWrite(timer0, 250000, false);         // sets how many tics will you count to
    trigger the interrupt (0.25 sec)
  timerAlarmEnable(timer0); // Enables timer
}
```

```
75
// Timer Interrupt for BLDC Motor Actuation
77 hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;
79 void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
81   motortimer = false; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
83 }
void TimerInterruptInit1(int duration) {  //The timer simply counts the number of Tic
    generated by the quartz. With a quartz clocked at 80MHz, we will have 80,000,000 Tics.
85   portENTER_CRITICAL_ISR(&timerMux1);
  motortimer = true;
87   portEXIT_CRITICAL_ISR(&timerMux1);
  timer1 = timerBegin(1, 80, true); // divides the frequency by the prescaler: 80,000,000 /
    80 = 1,000,000 tics / sec
89   timerAttachInterrupt(timer1, &onTime1, true);    // sets which function do you want to
    call when the interrupt is triggered
  timerAlarmWrite(timer1, duration, false);         // sets how many tics will you count to
    trigger the interrupt (0.25 sec)
91   timerAlarmEnable(timer1); // Enables timer
}
93


95 void setup() {
  pinMode(POT, INPUT);
97
  pinMode(LED_PIN, OUTPUT);
99   digitalWrite(LED_PIN, LOW); // LED initially off

101   pinMode(BTN_1, INPUT);
  attachInterrupt(BTN_1, isr_1, RISING);
103   pinMode(BTN_2, INPUT);
  attachInterrupt(BTN_2, isr_2, RISING);
105
  Serial.begin(115200);
107
  ledcSetup(ledChannel_1, freq, resolution);
109   ledcAttachPin(PWM_1, ledChannel_1);

111   Wire.begin();
  mpu.setup(0x68);  // need specific device I2C address
113 }


115
void loop() {
117   // put your main code here, to run repeatedly:
  if (CheckForONOFFButtonPress()) {
119     ONOFFButtonResponse();
  }
121   switch (state) {
    case 0: // idle
123       ledcWrite(ledChannel_1, 200); // idle signal to keep ESC engaged
      break;
125     case 1: // armed
      ledcWrite(ledChannel_1, 200); // arming signal, motor should NOT spin
127       if (CheckForJUMPButtonPress()) { // JUMP button only relevant when in armed state
        JUMPButtonResponse();
129       }
      break;
131     case 2: // jumping
      if (motortimer) {
133         ledcWrite(ledChannel_1, 500); // motor at MAX throttle
      }
135       else {
        digitalWrite(LED_PIN, LOW);
137         state = 0; // go back to idle when actuation time done
        motortimer = true;
139       }
      break;
141     default:
      Serial.print("STATE ERROR");
143       break;
  }
145   Serial.println(state);
}
147
```

```
// Event Checker for ON/OFF Button
bool CheckForONOFFButtonPress() {
  return ON_OFF_pressed;
}
// Event Checker for JUMP Button
bool CheckForJUMPButtonPress() {
  return JUMP_pressed;
}
// Event Checker to see if robot is UPRIGHT
bool CheckForUpright() {

  bool repeat = true;
  float AX;
  float AY;
  float AZ;
  while (repeat) { // code loops until update received from IMU (very small delay)
    if (mpu.update()) {
      AX = mpu.getAccX() - x_bias;
      AY = mpu.getAccY() - y_bias;
      AZ = mpu.getAccZ() - z_bias;
      repeat = false;
    }
  }
  return (pow(pow(AX, 2) + pow(AY, 2) + pow(AZ, 2), 0.5) <= 1);
    // will be TRUE if UPRIGHT (norm of acceleration vector within threshold)
}


// Service Function for ON/OFF Button
void ONOFFButtonResponse() {
  ON_OFF_pressed = false;
  TimerInterruptInit0(); // timer is NOT auto-reloading
  switch (state) {
    case 0: // idle --> armed
      digitalWrite(LED_PIN, HIGH);
      state = 1;
      break;
    case 1: // armed --> idle
      digitalWrite(LED_PIN, LOW);
      state = 0;
      break;
    case 2: // jumping --> idle
      digitalWrite(LED_PIN, LOW);
      state = 0;
      break;
    default:
      Serial.println("STATE ERROR");
      break;
  }
}
// Service Function for JUMP Button
void JUMPButtonResponse() {
  JUMP_pressed = false;
  TimerInterruptInit0(); // timer is NOT auto-reloading
  if (CheckForUpright() && (state == 1)) { // needs to be UPRIGHT and in JUMPING state
    state = 2; // armed --> jumping
    int duration = map(analogRead(POT), 0, 4095, 500000, 1000000);
    TimerInterruptInit1(duration); // set motor timer
    //Serial.println(duration);
  }
  else {
    return; // do nothing
  }
}
```

Appendix D: Grashof Analysis of Key Linkage Elements