Appendix I

Arduino State Machine

```
#define tu 125

byte Legs[] = {6, 7, 8, 9, 10, 11, 12, 13};
byte n_Legs = sizeof(Legs)/sizeof(byte);
byte LF1, LF2, LH1, LH2, RH2, RH1, RF2, RF1;
int prev_state = 5;

/* Setup up serial communication and digital
output pins */
void setup() {
    Serial.begin(9600);
    for (byte i = 0; i < n_Legs; i++) {
        pinMode(Legs[i], OUTPUT);
    }
}

/* Set pin n */
void hi(byte n) {
    digitalWrite(n, HIGH);
}

/* Clear pin n */
void lo(byte n) {
    digitalWrite(n, LOW);
}

/* Moves quadruped in a straight line
Values of state:
0 - relative forward
1 - relative left
2 - relative backward
3 - relative right
Cycle takes 10 time units */
void move_forward(int state) {
    if (state != prev_state) {
        prev_state = state;
        switch (state) {
            case 0:
                LF1 = Legs[0];
                LF2 = Legs[1];
                LH1 = Legs[2];
```

```
42                  LH2 = Legs[3];
43                  RH2 = Legs[4];
44                  RH1 = Legs[5];
45                  RF2 = Legs[6];
46                  RF1 = Legs[7];
47              case 1:
48                  LF1 = Legs[2];
49                  LF2 = Legs[3];
50                  LH1 = Legs[4];
51                  LH2 = Legs[5];
52                  RH2 = Legs[6];
53                  RH1 = Legs[7];
54                  RF2 = Legs[0];
55                  RF1 = Legs[1];
56              case 2:
57                  LF1 = Legs[4];
58                  LF2 = Legs[5];
59                  LH1 = Legs[6];
60                  LH2 = Legs[7];
61                  RH2 = Legs[0];
62                  RH1 = Legs[1];
63                  RF2 = Legs[2];
64                  RF1 = Legs[3];
65              default:
66                  LF1 = Legs[6];
67                  LF2 = Legs[7];
68                  LH1 = Legs[0];
69                  LH2 = Legs[1];
70                  RH2 = Legs[2];
71                  RH1 = Legs[3];
72                  RF2 = Legs[4];
73                  RF1 = Legs[5];
74          }
75      }
76      // 0 tu
77
78      // LF, RH in phase I
79      // RF, LH in phase IV
80      hi(LF2);
81      hi(RH2);
82      lo(RF1);
83      lo(LH1);
84
85      delay(tu);
86      // 1 tu
```

```
87
88      // LF, RH in phase II
89      hi(LF1);
90      hi(RH1);
91
92      delay(tu);
93      delay(tu);
94      // 3 tu
95
96      // LF, RH in phase III
97      lo(LF2);
98      lo(RH2);
99
100     delay(tu);
101     delay(tu);
102     // 5 tu
103
104     // LF, RH in phase IV
105     // RF, LH in phase I
106     lo(LF1);
107     lo(RH1);
108     hi(RF2);
109     hi(LH2);
110
111     delay(tu);
112     // 6 tu
113
114     // RF, LH in phase II
115     hi(RF1);
116     hi(LH1);
117
118     delay(tu);
119     delay(tu);
120     // 8 tu
121
122     // RF, LH in phase III
123     lo(RF2);
124     lo(LH2);
125
126     delay(tu);
127     delay(tu);
128     // 10 tu
129     // end of cycle
130 }
131
```

```
/* Grasp with 4 channels in 2 diametrically
opposed legs */
void grasp() {
    prev_state = 4;
    hi(Legs[0]);
    hi(Legs[1]);
    hi(Legs[4]);
    hi(Legs[5]);
}

/* Test each channel by inflating and deflating 4 time
units each */
void test() {
    prev_state = 5;
    for (int i = 0; i < n_Legs; i++) {
        hi(Legs[i]);
        delay(tu);
        delay(tu);
        delay(tu);
        delay(tu);
        lo(Legs[i]);
        delay(tu);
        delay(tu);
        delay(tu);
        delay(tu);
    }
}

/* Close all channels */
void halt() {
    prev_state = 6;
    for (int i = 0; i < n_Legs; i++) {
        lo(Legs[i]);
    }
}

/* State machine */
void handle(int state) {
    if (state >= 0 && state <= 3) move_forward(state);
    else if (state == 4) grasp();
    else if (state == 5) test();
    else halt();
}

/* Main loop, polls for serial communication, then
```

```
177 invokes state handler */
178 void loop() {
179     if (Serial.available()) {
180         int state = Serial.parseInt();
181         handle(state);
182     }
183 }
```