```
 1
 2  #define COLOR_HUE_FORWARD   0.0
 3  #define COLOR_HUE_LEFT   0.25
 4  #define COLOR_HUE_BACKWARD 0.5
 5  #define COLOR_HUE_RIGHT  0.75
 6
 7  #define CLAV_BRAIN_TURN_SPEED_LEFT  40
 8  #define NINETY_DEGREE_LEFT  69//67  //Right ticks to turn left
 9
10  #define CLAV_BRAIN_TURN_SPEED_RIGHT 40
11  #define NINETY_DEGREE_RIGHT 69  //Left ticks to turn right //66 //70
12
13  #define PING_TARGET_CM_STOP   60 //We can see two boxes away!
14  #define PING_TARGET_CM_MOVE   24
15  //#define PING_TARGET_MM   500
16
17  #define INVESTIGATE_ITERATES    2
18
19  #define WANDER_TARGET_COL  4
20  #define WANDER_TARGET_ROW  4
21
22  #define GO_HOME_TARGET_COL  0
23  #define GO_HOME_TARGET_ROW  0
24
25  #define INCLINE_TARGET_COL  -5
26  #define INCLINE_TARGET_ROW  0
27
28  #define INCLINE_TARGET_THETA  ONE_PI
29
30  #define INCLINE_SPEED_FAST 16
31  #define INCLINE_SPEED_SLOW 8
32  #define COLLISION_SPEED   5
33  #define WANDER_SPEED      10
34  #define KNOWN_SPEED       15
35
36  int targetCol, targetRow;
37  boolean parity;
38
39  //=========================================================================================
40  // Initializer
41  //=========================================================================================
42  inline void clavInit(){
43    tickCountLeft = tickCountRight = 0;
44    moveState = STOPPED;
45    updateDirection();
46
47    specificColor(COLOR_HUE_FORWARD);
48
49    dTheta = 0;
50    dX = 0;
51    dY = 0;
52
53    x = EXCESS / 2; //0;
54    y = EXCESS / 2; //0;//200;//50.0;//80.0;
55    thetaState = PI_O2;
56    theta = radianToValue(thetaState);
57    addShift(thetaState);
58
59    targetCol = WANDER_TARGET_COL;
60    targetRow = WANDER_TARGET_ROW;
61
62    radianPanTo( RANGER_FORWARD );
63    delay(50);
64    parity = false;
65    //anglePanTo(90);
66    //Straight( 10, 1 );
67  }
68
69  //=========================================================================================
```

```
 70   // Loop
 71   //========================================================================================
 72   inline void clavBrain()
 73   {
 74
 75     //if(moveState == FORWARD)
 76       //pingCheck();
 77
 78     //delay(100);
 79
 80     //---- update robot config (x,y,theta)
 81     dX = PI * WHEEL_RADIUS * cos(theta) * ((double)(tickCountLeft + tickCountRight) /
     TICK_PER_ROT);
 82     xNoSkew = xNoSkew + dX;
 83     x = x + dX;
 84
 85     dY = PI * WHEEL_RADIUS * sin(theta) * ((double)(tickCountLeft + tickCountRight) /
     TICK_PER_ROT);
 86     yNoSkew = yNoSkew + dY;
 87     y = y + dY;
 88
 89
 90     tickCountLeft = tickCountRight = 0;
 91
 92     markCurrentPass();
 93
 94     #ifdef DEBUG
 95     Serial.print(x); Serial.print(" :x || y: "); Serial.println(y); Serial.print("\t");
 96     Serial.print(valueToSectorMM( x )); Serial.print(" :col || row: "); Serial.println
     (valueToSectorMM( y ));
 97     #endif
 98
 99     //Change our light
100     if(parity)
101       radianToColor( radianToValue( moveEffectTheta() ) );
102     else
103       radianToColor( theta );
104
105     parity = !parity;
106
107     if(machineState == WANDER || machineState == WANDEROUT){
108       clavGoPlace(true);
109       //---- check if we're completely in the
110       if ( posInTarget(targetCol, targetRow) )
111       {
112         ClavStop();
113
114         #ifdef DEBUG
115         Serial.println("======================");
116         Serial.println("\t Wander Target Reached!");
117         Serial.println("======================");
118         #endif
119
120         //---- update state
121         radianPanTo( RANGER_FORWARD );
122         delay(100);
123
124         machineState = GOHOME;
125         targetCol = GO_HOME_TARGET_COL;
126         targetRow = GO_HOME_TARGET_ROW;
127       }
128     }
129     else if (machineState == GOHOME)
130     {
131       //---- use map to go home
132       clavGoPlace(false);
133
134       //---- check if we're completely in the
135       if ( posInTarget(targetCol, targetRow) )
```

```
136       {
137         ClavStop();
138
139         #ifdef DEBUG
140         Serial.println("======================");
141         Serial.println("\t Go Home - Target Reached!");
142         Serial.println("======================");
143         #endif
144
145         machineState = INCLINE;
146         targetCol = INCLINE_TARGET_COL;
147         targetRow = INCLINE_TARGET_ROW;
148
149         turnToTheta(ONE_PI);
150
151         lightCycle(50);
152       }
153     }
154     else if (machineState == INCLINE) {
155       if ( getDiodeVal() > 100) {
156         if (lightState == LEFT) {
157           //Set Right
158           MOTOR.setSpeedDir1(INCLINE_SPEED_FAST, DIRF);
159           //Set Left
160           MOTOR.setSpeedDir2(INCLINE_SPEED_SLOW, DIRR);
161           lightState = RIGHT;
162           delay(100);
163         }
164         else if (lightState == RIGHT){
165           //Set Right
166           MOTOR.setSpeedDir1(INCLINE_SPEED_SLOW, DIRF);
167           //Set Left
168           MOTOR.setSpeedDir2(INCLINE_SPEED_FAST, DIRR);
169           lightState = LEFT;
170           delay(100);
171         }
172       }
173
174       //---- check if we're completely in the
175       if ( x <= -1550 )
176       {
177         ClavStop();
178
179         #ifdef DEBUG
180         Serial.println("======================");
181         Serial.println("\t Incline - Target Reached!");
182         Serial.println("======================");
183         #endif
184
185         machineState = DONE;
186       }
187     }
188     else if (machineState == DONE){
189       lightCycle(100);
190     }
191   }
192
193   //~~~~~~~~~~~~~
194   // Sub-brain: Wander (for wandering, and the like)
195   //~~~~~~~~~~~~~
196   void clavGoPlace(boolean investigate){
197
198     //If we've stopped
199     if( moveState == STOPPED ){
200       int speedWeDo;
201       //Investigate the surrounding area
202       if(investigate){
203         investigatePing();
204         speedWeDo = WANDER_SPEED;
```

```
205       }
206       else
207         speedWeDo = KNOWN_SPEED;
208
209       #ifdef DEBUG
210       printPassGrid();
211       delay(100);
212       printStatGrid();
213       delay(100);
214       #endif
215
216       //Formulate a plan
217       pathFindFromCurrent(targetCol, targetRow);
218
219       tickCountLeft = tickCountRight = 0;
220
221       if(investigate){
222         radianPanTo( RANGER_FORWARD ); delay(50);
223       }
224
225       #ifdef DEBUG
226       Serial.println( getStartAction() );
227       #endif
228
229       //Follow the plan
230       switch( getStartAction() ){
231         default:
232         case ACTION_STOP:
233         case ACTION_NONE:
234           break;
235
236         case ACTION_FORWARD:
237           ClavForward(speedWeDo);break;
238
239         case ACTION_REVERSE:
240           ClavReverse(speedWeDo); break;
241
242         case ACTION_LEFT_FORWARD:
243           TurnLeft90(); ClavForward(speedWeDo); break;
244
245         case ACTION_RIGHT_FORWARD:
246           TurnRight90(); ClavForward(speedWeDo); break;
247       }
248     }
249     //If we've reached our subtarget
250     else if( posInSubTarget() ){
251       //Stop!
252       ClavStop();
253       #ifdef DEBUG
254       Serial.println("======================");
255       Serial.println("\t Subtarget Reached!");
256       Serial.println("======================");
257       #endif
258     }
259     // If we're moving and we're suddenly attacked by a wild box
260     else if( moveState == FORWARD && digitalRead(buttonPin) )
261     {
262       //---- stop
263       ClavStop();
264       //---- mark that we had a collision, adjust our sights
265       blockCollision();
266
267       tickCountLeft = tickCountRight = 0;
268
269       ClavReverse(COLLISION_SPEED);
270
271       #ifdef DEBUG
272       Serial.println("======================");
273       Serial.println("\t Button Pressed!");
```

```
274       Serial.println("======================");
275       #endif
276     }
277     //otherwise, if we're moving forward and we go out of bounds
278     else if( (moveState == FORWARD || moveState == BACKWARD) && outOfBoundry() &&
      machineState == WANDER )//(OUT_OF_BOUNDRY) )
279     {
280       int oldState = moveState;
281       //---- stop
282       ClavStop();
283
284       //Mark that we've gone out of bounds
285       machineState = WANDEROUT;
286
287       #ifdef DEBUG
288       Serial.println("======================");
289       Serial.println("\tOut of Bounds!");
290       Serial.println("======================");
291       #endif
292
293       if(oldState == FORWARD)
294         ClavReverse(WANDER_SPEED);
295       else if(oldState == BACKWARD)
296         ClavForward(WANDER_SPEED);
297     }
298     else if ( machineState == WANDEROUT && inBoundry() )//(IN_BOUNDRY))
299     {
300       //---- stop
301       ClavStop();
302       machineState = WANDER;
303     }
304     /*
305     else if(machineState == WANDER || machineState == WANDEROUT){
306       projectionCheck();
307     }
308     */
309   }
310
311   //============================================================================================
312   // Initiate the forward - this code is duplicated so much that we might as well
313   //============================================================================================
314   inline void ClavForward(int inSpeed) {
315     //---- update state
316     moveState = FORWARD;
317     updateDirection();
318     //---- go forward
319     forward(inSpeed);
320     //---- R E S T    A E S T H E T I C A L L Y
321     delay(100);
322   }
323
324   //============================================================================================
325   // Initiate the reverse - this code is duplicated so much that we might as well
326   //============================================================================================
327   inline void ClavReverse(int inSpeed) {
328     //---- update state
329     moveState = BACKWARD;
330     updateDirection();
331     //---- back up
332     backward(inSpeed);
333     //---- R E S T    A E S T H E T I C A L L Y
334     delay(100);
335   }
336
337   //============================================================================================
338   // Stop the car - this code is duplicated so much that we might as well
339   //============================================================================================
340   inline void ClavStop() {
341     //---- stop
```

```
342     stopMotion();
343     //---- R E S T    A E S T H E T I C A L L Y
344     delay(100);
345     //---- update state
346     moveState = STOPPED;
347     updateDirection();
348   }
349
350   //=============================================================================================
351   // TurnLeft90
352   //=============================================================================================
353   void TurnLeft90() {
354       //Set up for turning left
355       specificColor(COLOR_HUE_LEFT);
356       moveState = TURN_LEFT;
357       updateDirection();
358       tickCountLeft = tickCountRight = 0;
359
360       //begin the turn
361       turnInPlaceLeft(CLAV_BRAIN_TURN_SPEED_LEFT);
362
363       //while loop to turn as we please
364       while (tickCountRight < NINETY_DEGREE_LEFT)
365       {
366         delayMicroseconds(1);
367       }
368
369       //Stop the turn
370       ClavStop();
371
372       #ifdef DEBUG
373       Serial.println("======================");
374       Serial.println("\t Turning Left!");
375       Serial.println("======================");
376       #endif
377
378       tickCountLeft = tickCountRight = 0;
379
380       removeShift(thetaState);
381
382       //Update our theta (positively)
383       incrementTheta();
384
385       addShift(thetaState);
386
387       //R E S T    A E S T H E T I C A L L Y
388       delay(100);
389   }
390
391   //=============================================================================================
392   // TurnRight90
393   // dirn is 1 for right, -1 for left
394   //=============================================================================================
395   void TurnRight90() {
396       //Set up for turning right
397       specificColor(COLOR_HUE_RIGHT);
398       moveState = TURN_RIGHT;
399       updateDirection();
400       tickCountLeft = tickCountRight = 0;
401
402       //begin the turn
403       turnInPlaceRight(CLAV_BRAIN_TURN_SPEED_RIGHT);
404
405       //while loop to turn as we please
406       while (tickCountLeft < NINETY_DEGREE_RIGHT)
407       {
408         delayMicroseconds(1);
409       }
410
```

```
411       //Stop the turn
412       ClavStop();
413
414       #ifdef DEBUG
415       Serial.println("======================");
416       Serial.println("\t Turning Right!");
417       Serial.println("======================");
418       #endif
419
420       tickCountLeft = tickCountRight = 0;
421
422       removeShift(thetaState);
423
424       //Update our theta (negatively)
425       decrementTheta();
426
427       addShift(thetaState);
428
429       //R E S T    A E S T H E T I C A L L Y
430       delay(100);
431   }
432
433   //================================================================================
434   // TurnToTheta
435   //================================================================================
436   void turnToTheta(int target){
437     switch( target - thetaState ){
438       case -1: TurnRight90();
439         break;
440       case 1: TurnLeft90();
441         break;
442       case 2:
443       case -2: TurnRight90(); TurnRight90();
444         break;
445       default:
446       case 0:
447         break;
448     }
449   }
450
451   //Take a look around, see?
452   void investigatePing(){
453     //start left
454     int angle = RANGER_LEFT;
455
456     //for each angle
457     for(int i = 0; i < 3; i++){
458       //Pan to that angle
459       radianPanTo(angle);
460       delay(50);
461
462       //If whatever we're scanning isn't automatically going to be out of bounds
463       if( inSectors(valueToSectorMM( x ) + projectColPan(), valueToSectorMM( y ) +
      projectRowPan()) ){
464         //Then ping X times
465         for(int j = 0; j < INVESTIGATE_ITERATES; j++){
466           pingCheck();
467           delay(100);
468         }
469       }
470       angle = radianLeft(angle);
471     }
472     evalPanTheta();
473   }
474
475   inline void pingCheck(){
476     boolean test;
477     int dist_cm = pingCM();
478     double objX, objY;
```

```
479     int objRow, objCol;
480
481     if( pingMoveCheck(dist_cm) || pingStopCheck(dist_cm) ) {
482       objX = (x / 10) + cos( evalPanTheta() ) * dist_cm;
483       objY = (y / 10) + sin( evalPanTheta() ) * dist_cm;
484
485       objCol = valueToSectorCM( objX );
486       objRow = valueToSectorCM( objY );
487
488       if( objCol == valueToSectorMM( x ) && objRow == valueToSectorMM( y ) ){
489           objCol += projectColPan();
490           objRow += projectRowPan();
491       }
492
493       if( inSectors( objCol, objRow ) )
494       {
495         if( getSectorStat( objCol, objRow ) <= 0.0 )
496           setSectorStat( objCol, objRow, 0.5);
497
498         incrementSectorStat( objCol, objRow, ( 1.0 - getSectorStat( objCol, objRow ) ) / 2.0
    );
499       }
500     }
501   }
502
503   inline boolean pingMoveCheck(int dist_cm){
504     return (moveState == FORWARD || moveState == BACKWARD) && dist_cm < PING_TARGET_CM_MOVE;
505   }
506
507   inline boolean pingStopCheck(int dist_cm){
508     return moveState == STOPPED && dist_cm < PING_TARGET_CM_STOP;
509   }
```