

a routine for a walker with 6 sensors:
2 sensors in front called : LEFT0, RIGHT0
2 sensors on left side called : LEFT1, LEFT2
2 sensors on right side called : RIGHT1, RIGHT2

The other variables refer to defined distances registered by a sensor to act as thresholds for different motor activations.
Too close in front : stopT

Too close on side : sidestopT
Too far on side : searchT
Ideal distance on side : target

The user settings refer to the
brake button : buttonpin
anterior-posterior joystick: JOYSTICKAP
medial-lateral joystick : JOYSTICKML

There was an earlier weirder version I made too. Does this match with what Paul showed you? Does it make sense?

```
// STOP:
// if both front sensors are "too close" or brake is set
if LEFT0 < stopT OR RIGHT0 < stopT OR buttonpin==1
{
  // stop
}
else

  // check for obstacle to avoid:
  // if one front sensor is within avoid range
  if (LEFT0 < turnT & LEFT0 > stopT) OR (RIGHT0 < turnT & RIGHT0 > stopT)
  { if LEFT0 < RIGHT0 // if left is closer
    { // turn right (slow right motor) }
    else // if right is closer
    { // turn left (slow left motor) }
  }

  // check for walls:
  // if both side sensors are within search range
  if RIGHT1 > sidestopT & RIGHT1 < searchT & RIGHT1 > sidestopT & RIGHT1 < searchT
  { if RIGHT1 < target // too far from target wall distance
    { // turn right (slow right motor) }
    else // too close to target wall distance
    { // turn left (slow left motor) }
  }
  if LEFT1 > sidestopT & LEFT1 < searchT & LEFT2 > sidestopT & LEFT2 < searchT
  { if LEFT1 < target // too far from target wall distance
    { // turn left (slow left motor) }
    else // too close to target wall distance
    { // turn right (slow right motor) }
  }

  // check user controls

  // USER: forward
  if JOYSTICKAP== forward & JOYSTICKAP==0
  { // forward }

  // USER: left
  if JOYSTICKML== left
  { // left (slow left motor)}
  // USER: right
  if JOYSTICKML== right
  { // right (slow right motor)}
```