1. Hammer header on Pi - **15m**
2. Physical connections and boot - **15m**
3. Initial setup & connect to wifi - **5m**
4. Students accidentally hit update and we take a tour of SVSU **45m**
5. Install Joy Bonnet & Restart **10m**
   1. Terminal
      1. curl https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/joy-bonnet.sh >joy-bonnet.sh
      2. sudo bash joy-bonnet.sh
      3. Yes, disable overscan
      4. No, do not install gpis-halt utility
      5. Yes, continue
      6. Wait
      7. Yes, reboot
6. Configure Joy Bonnet - **15m**
   1. Terminal
      1. sudo nano /boot/joyBonnet.py
      2. edit file using nano to this effect (follow file’s syntax)  
         BUTTON\_A: e.KEY\_M,  
         BUTTON\_B: e.KEY\_N,  
         BUTTON\_X: e.KEY\_K,  
         BUTTON\_Y: e.KEY\_J,  
         SELECT: e.KEY\_LEFTCTRL,  
         START: e.KEY\_ENTER,  
         PLAYER1: e.KEY\_1,  
         PLAYER2: e.KEY\_2,  
         1000: e.KEY\_W,  
         1001: e.KEY\_S,  
         1002: e.KEY\_A,  
         1003: e.KEY\_D
      3. Save using Ctrl + o
         1. do not modify filename, hit enter
      4. Exit using Ctrl + x
   2. Reboot
7. Test Joy Bonnet - **5m**
   1. Terminal
      1. Press the buttons to make sure they line up with what they’re supposed to do.
8. Game Build -
   1. Create a folder in /home/pi called “snake”
   2. Create a new file inside of “snake” called “snake.py”
   3. Open “snake.py” with Thonny Python Editor
      1. Imports - We need to import some already made files to use to build our game
         1. Code:  
            import pygame  
            import sys  
            import random  
            import time
      2. Creating the snake - We now need to create a class (or blueprint) for our snake to be built by the computer. Right now we’re going to just make the snake start in the same place always.
         1. Code:  
            class Snake():  
             def \_\_init\_\_(self):  
             self.position = [100,50]  
             self.body = [[100,50],[90,50],[80,50]]  
             self.direction  
             self.direction = “RIGHT"  
             self.changeDirectionTo = self.direction
      3. Now we need to tell the snake that it can turn, but it is not allowed to go straight backwards. (point out assignment operator vs is equal to )
         1. Code:  
            def.changeDirTo(self,dir):  
             if dir==“RIGHT” and not self.direction==“LEFT”:  
             self.direction = “RIGHT”  
             if dir==“LEFT” and not self.direction==“RIGHT”:  
             self.direction = “LEFT”  
             if dir==“DOWN” and not self.direction==“UP”:  
             self.direction = “DOWN”  
             if dir==“UP” and not self.direction==“DOWN”:  
             self.direction = “UP”
      4. Now we need actually move the snake along and check to see if it hits a piece of food as it is moving and what to do when that happens. (explain that x += y is the same as x = x + y where it is assigning the new value to x on the left of the equals sign)
         1. Code:  
            def move(self,foodPos):  
             if self.direction == “RIGHT”:  
             self.position[0] += 10  
             if self.direction == “LEFT”:  
             self.position[0] -= 10  
             if self.direction == “UP”:  
             self.position[1] += 10  
             if self.direction == “DOWN”:  
             self.position[1] -= 10  
             self.body.insert(0,list(self.position))  
             if self.position == foodPos:  
             return 1  
             else:  
             self.body.pop()  
             return 0
      5. Now we need to make sure that the snake does not leave the boundaries of the screen.
         1. Code:  
            def checkCollision(self):  
             if self.position[0] > 490 or self.position[0] < 0:  
             return 1  
             elif self.position[1] > 490 or self.position[1] < 0:  
             return 1
      6. Now we need to make sure that the snake can not run into itself
         1. Code:  
            for bodyPart in self.body[1:]:  
             if self.position == bodyPart:  
             return 1  
            return 0
      7. Now we need to establish where the snake actually is per frame
         1. Code:  
            def getHeadPos(self):  
             return self.position  
            def getBody(self):  
             return self.body
      8. Let’s create a food generator that places food at random on the screen
         1. Code:  
            class FoodSpawner():  
             def \_\_init\_\_(self):  
             self.position = [random.randrange(1,50)\*10, random.randrange(1,50)\*10]  
             self.isFoodOnScreen = True  
             def spawnFood(self):  
             if self.isFoodOnScreen == False:  
             self.position = [random.randrange(1,50)\*10, random.randrange(1,50)\*10]  
             self.isFoodOnScreen = True  
             return self.position
      9. Now that the food shows up, we need to implement a way to add more food when the snake eats the food we just generated.
         1. Code:  
            def setFoodOnScreen(self,b):  
             self.isFoodOnScreen = b
      10. Now that we have the rules for the snake and the rules for the food, let’s put them together with some gameplay. We will start by creating a window and setting how fast the game renders.
          1. Code:  
             window = pygame.display.set\_mode((500,500))  
             pygame.display.set\_caption(“Snake”)  
             fps = pygame.time.Clock()
      11. Okay, so we have a window. Let’s set a score.
          1. Code:  
             score = 0
      12. And now a hook to allow our snake and food to exist
          1. Code:  
             snake = Snake()  
             foodSpawner = FoodSpawner()
      13. Now we can try to run the game.
          1. The screen will just be black at this point, but a window will open. Help students work through syntax errors up to this point. The X button might not close the game. Move the window and click the stop button in Thonny.
      14. Now we need to add a mechanism to quit the game
          1. Code:  
             def gameOver():  
              pygame.quit()  
              sys.exit
      15. Now to tie it all together with the gameplay to move the snake
          1. Code:  
             while True:  
              for event in pygame.event.get():  
              if event.type == pygame.QUIT:  
              gameOver()  
              elif event.type == pygame.KEYDOWN:  
              if event.key == pygame.K\_d:  
              snake.changeDirTo(“RIGHT”)  
              if event.key == pygame.K\_s:  
              snake.changeDirTo(“DOWN”)  
              if event.key == pygame.K\_a:  
              snake.changeDirTo(“LEFT”)  
              if event.key == pygame.K\_w:  
              snake.changeDirTo(“UP”)
      16. Next is to make the food spawn and tell it how to act when the snake hits it.
          1. Code:  
             foodPos = foodSpawner.spawnFood()  
             if(snake.move(foodPos)==1):  
              score += 1  
              foodSpawner.setFoodOnScreen(False)
      17. Let’s change the background color
          1. Code:  
             window.fill(pygame.Color(255,0,255))
      18. Now we’ll put the snake on the screen
          1. Code:  
             for pos in snake.getBody():  
              pygame.draw.rect(window,pygame.Color(0,225,0),pygame.Rect(pos[0],pos[1],10,10))
      19. And now we put the food on the screen.
          1. Code:  
             pygame.draw.rect(window,pygame.Color(225,0,0),pygame.Rect(foodPos[0],foodPos[1],10,10))
      20. Now we check to see if the snake has collided with the food
          1. Code:  
             if(snake.checkCollision()==1):  
              gameOver()
      21. Set the title to include your score and tell the computer how to display the game the right way
          1. Code:  
             pygame.display.set\_caption(“Snake | Score : “+str(score) )  
             pygame.display.flip()  
             fps.tick(15)