CS 240: Programming in C

Lecture 26: Hardware Interfacing

PURDUE UNIVERSITY

Announcements

- The final exam is on 12/13 at 7:00 pm
  - In STEW 183
- Wednesday will be a review lecture
  - Come prepared with questions!
- Homework 13 is due this Wednesday at 9:00 pm!

Announcements

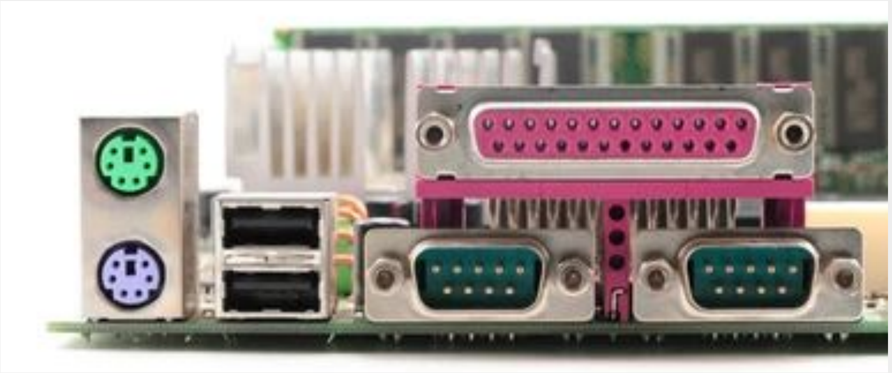
- Course Evaluations
  - Please complete them!
  - Survey closes on 12/8

Interfacing with hardware

- Writing software is fun, but sometimes we want to see how we can interact with the physical world
- Output
  - We can affect external state
- Input
  - We can read external state

Ports

- Computers used to have things like serial and parallel ports
  - Great for (slow) communication, easy to interface
  - Rarely exist now



Source: shutterstock.com

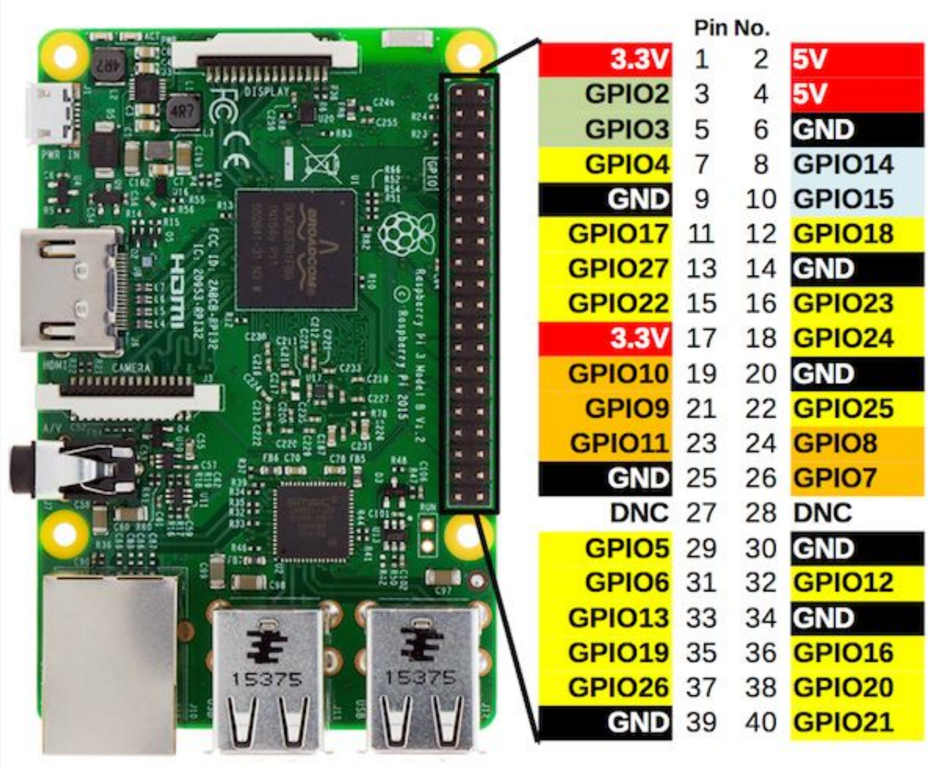
Ports

- Computers used to have things like serial and parallel ports
  - Great for (slow) communication, easy to interface
  - Rarely exist now
- Replaced by more modern ports
  - Ethernet port - need the ethernet protocols
  - USB port - difficult to use; can’t directly manipulate signal
- Look in /dev on Linux

Embedded systems

- System on a Chip (SoC)
    - Contains all/most computer components on a single board
    - CPU, memory, storage, ports, etc.
    - Many can run a full-fledged OS (e.g., Raspberry Pi)
- Often with exposed pins
    - General Purpose Input Output (GPIO)
    - Easy to interface

Raspberry Pi GPIO



Source: bigmessowires.com

Raspberry Pi GPIO

- Use pins to form a circuit with LEDs and buttons
  - Or any other components
- 3.3V and 5V power sources
- Ground pins
- GPIO pins can be accessed from a C program

Be careful!

- Incorrectly wiring the GPIO pins can permanently damage the device
- Use appropriate resistors
- Double check your circuit before you connect it to your RPi

Datasheet

- Sometimes called “spec sheet”
- Details technical characteristics of a component
  - Can be hardware or software
- Often includes
  - Functional descriptions
  - Pin diagrams
  - Voltage ratings and specs
  - Power consumption
  - I/O waveforms
  - Timings
  - Physical dimensions
  - etc.

BCM2711

- Raspberry Pi 4B uses a 1.5GHz quad core 64-bit ARM processor (BCM2711)
- Older Pis use BCM2835/6/7

Raspberry Pi GPIO access

- RPI.h
- pins.c
- pins.h
- traffic.c
- button.c

Function select

```
#define INP_GPIO(g) *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio.addr + ((g)/10)) |=  (1<<(((g)%10)*3))
```

- g is the pin number
- Each register controls 10 pins
  - 0-9, 10-19, 20-29, 30-39, 40-49, 50-57
- `(gpio.addr + ((g)/10))` gives us the address of the correct register for pin g

PURDUE UNIVERSITY

Function select

```
#define INP_GPIO(g) *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio.addr + ((g)/10)) |=  (1<<(((g)%10)*3))
```

- Say g = 3
- `((g)%10)*3` = 9
  - gives us the bit offset for that pin

pin 9 pin 8 pin 7 pin 6 pin 5 pin 4 pin 3 pin 2 pin 1 pin 0

```
00000000 00000000 00000000 00000000
```

Function select

```
#define INP_GPIO(g) *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio.addr + ((g)/10)) |=  (1<<(((g)%10)*3))
```

- Say g = 3
- `((g)%10)*3` = 9
  - gives us the bit offset for that pin

pin 9 pin 8 pin 7 pin 6 pin 5 pin 4 pin 3 pin 2 pin 1 pin 0

```
00000000 00000000 00000000 00000000
```

`7`	`00000000 00000000 00000000 00000111`

Function select

```
#define INP_GPIO(g) *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio.addr + ((g)/10)) |=  (1<<(((g)%10)*3))
```

- Say g = 3
- `((g)%10)*3` = 9
  - gives us the bit offset for that pin

pin 9 pin 8 pin 7 pin 6 pin 5 pin 4 pin 3 pin 2 pin 1 pin 0

```
00000000 00000000 00000000 00000000
```

`7 << 9`

```
00000000 00000000 00001110 00000000
```

Function select

```
#define INP_GPIO(g) *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio.addr + ((g)/10)) |=  (1<<(((g)%10)*3))
```

- Say g = 3
- `((g)%10)*3` = 9
  - gives us the bit offset for that pin

pin 9 pin 8 pin 7 pin 6 pin 5 pin 4 pin 3 pin 2 pin 1 pin 0

```
00000000 00000000 00000000 00000000
```

`~(7 << 9)`

```
11111111 11111111 11110001 11111111
```

GPIO Application

- Realtime Photometric Stereo
- Theory
  - Given 3+ images of a static scene (no camera/scene motion)
  - Each image has a different light source
  - You can solve for per-pixel “surface normals” (direction of surface)
  - With good normals, you can extrude a rough 3D surface

GPIO Application

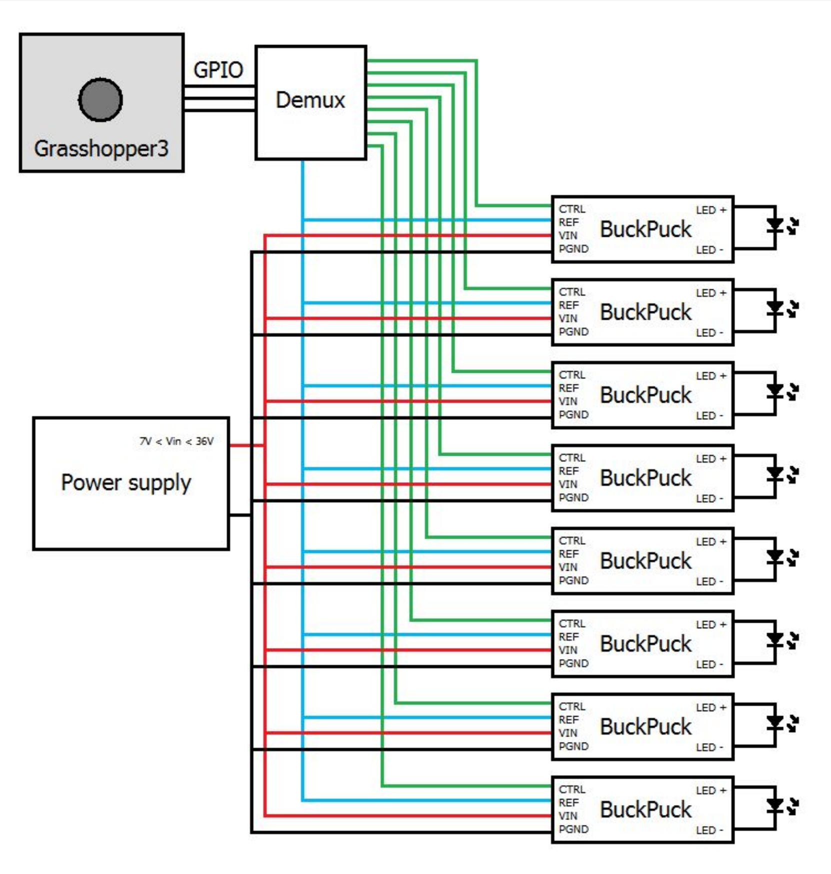
- Main idea:
  - High-speed camera
  - Array of high-power LEDs sync'd to framerate
  - Use GPU to recover normals quickly

PURDUE UNIVERSITY

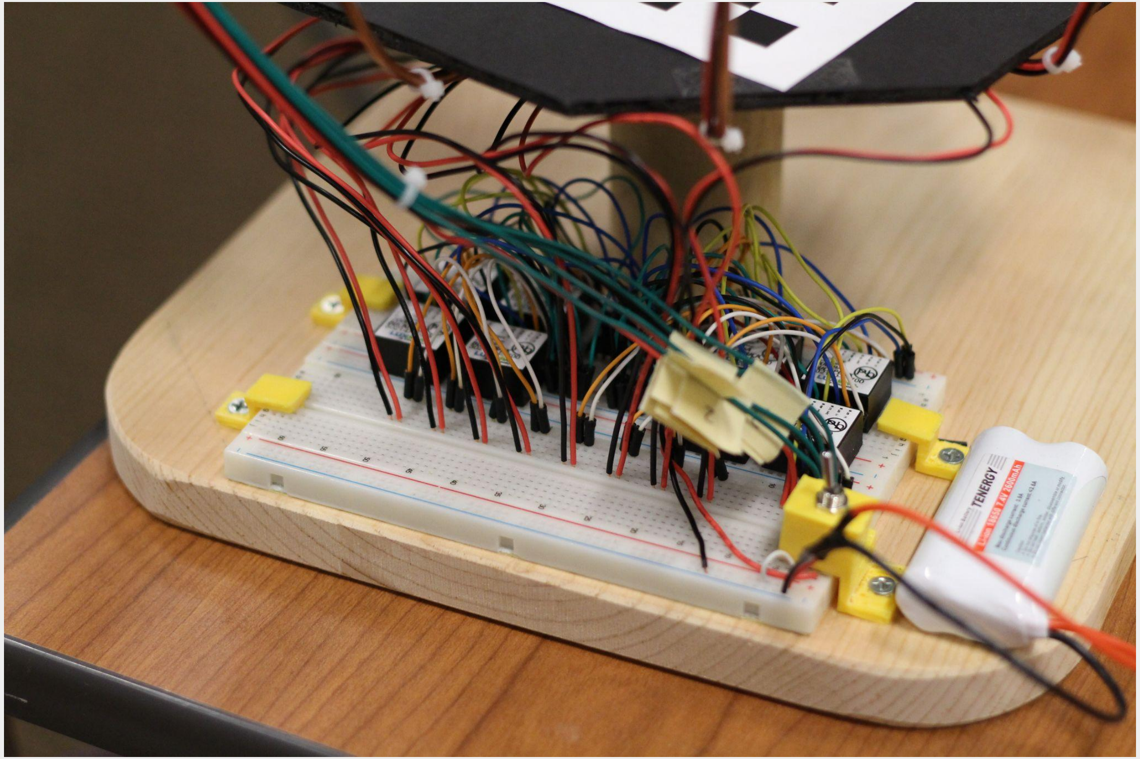
GPIO Application

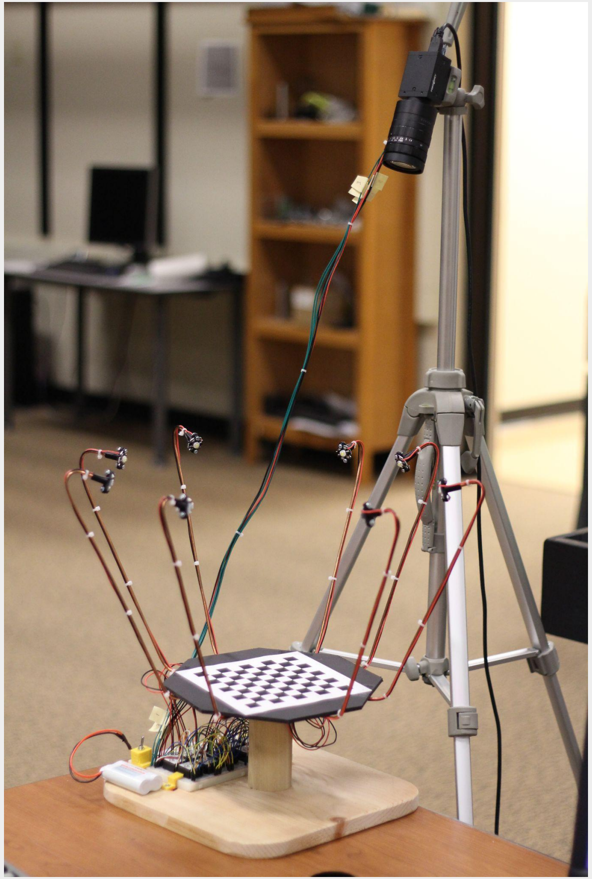
- How? The camera has GPIO pins!

GPIO Application

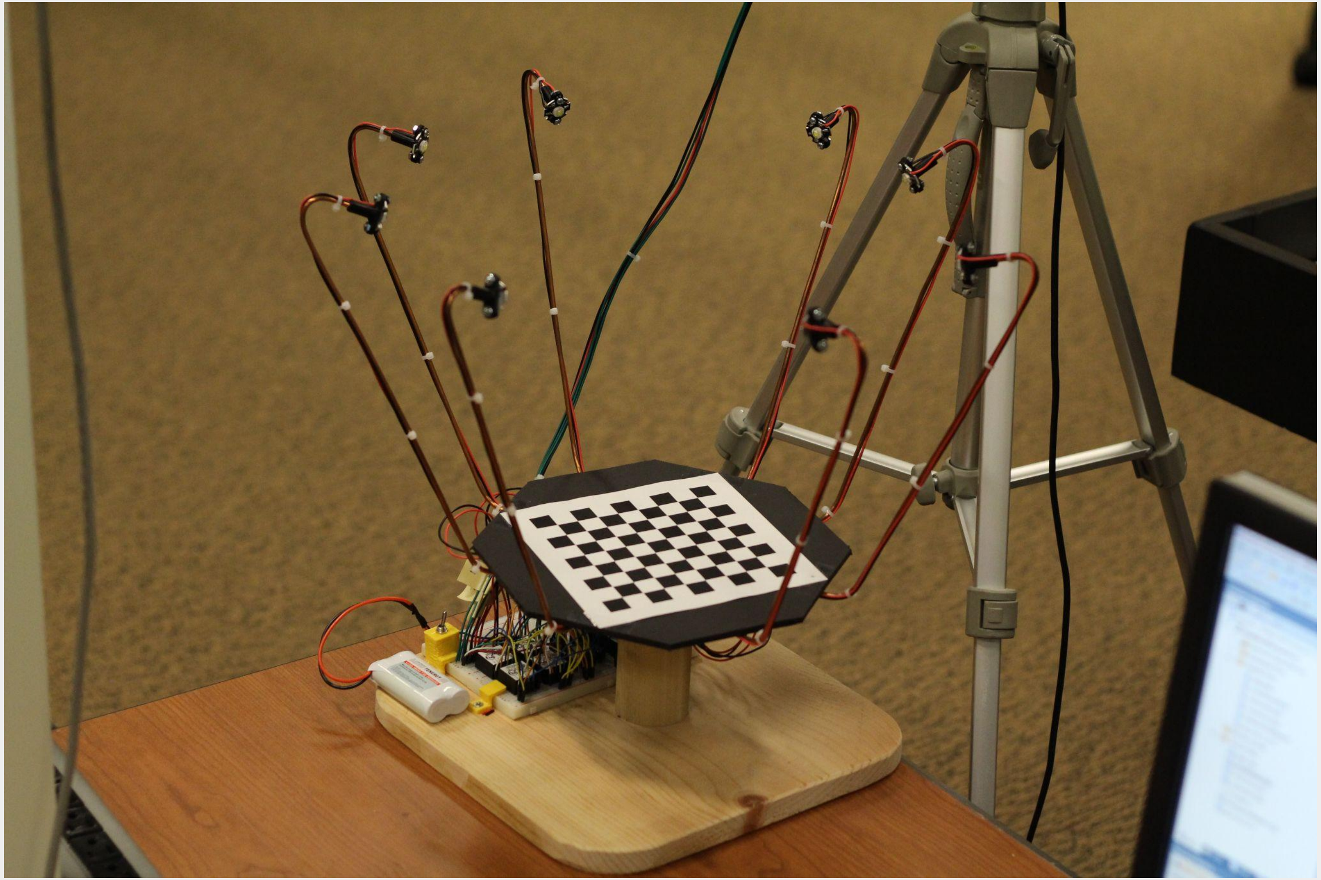


GPIO Application





GPIO Application



GPIO Application