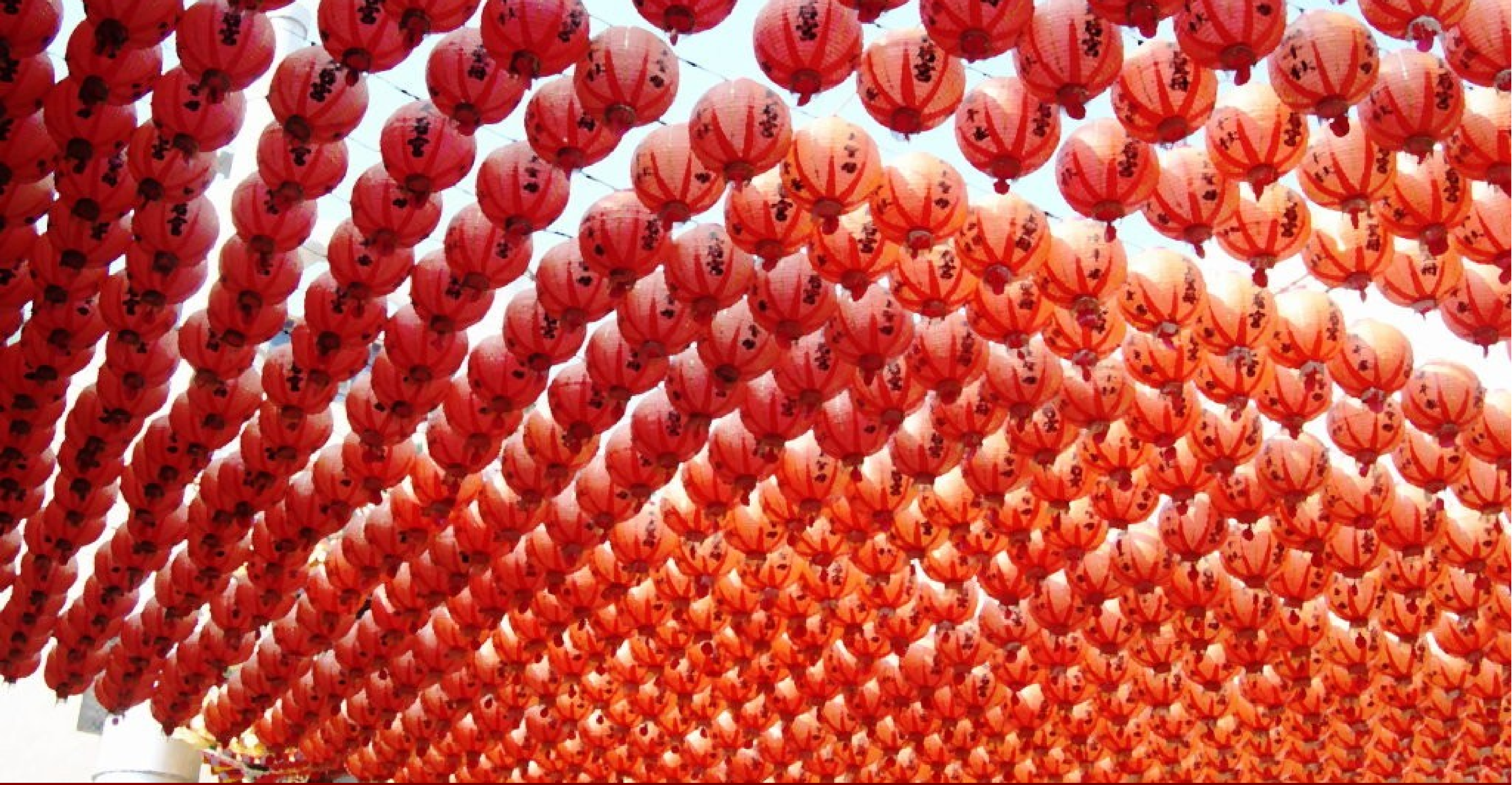


Non-premptive Multitasking for Arduino

Pete Soper, Apex Proto Factory
TriEmbed April 14, 2018

Outline

- Who Is This Guy?
- Context
- Composing Asynchronous Programs with Arduino
- The Right Tool for the Job
- A Simple Task Library
    - Characteristics
    - Key API
    - State Diagram
- References and Q & A

Who Is This guy?

- Auburn and University of Alabama, Huntsville
  - Started engineering, focus on experimental psych, found CS
  - Most effort put into human factors prep for Skylab and Shuttle: learned computing from the metal up as a side effect
- @OEMs writing language tool chain, virtualization, OS and datacomm software, ending with architecture
  - Data General, Business Application Systems, Network Products, Encore Computer, Sun Microsystems

Who?

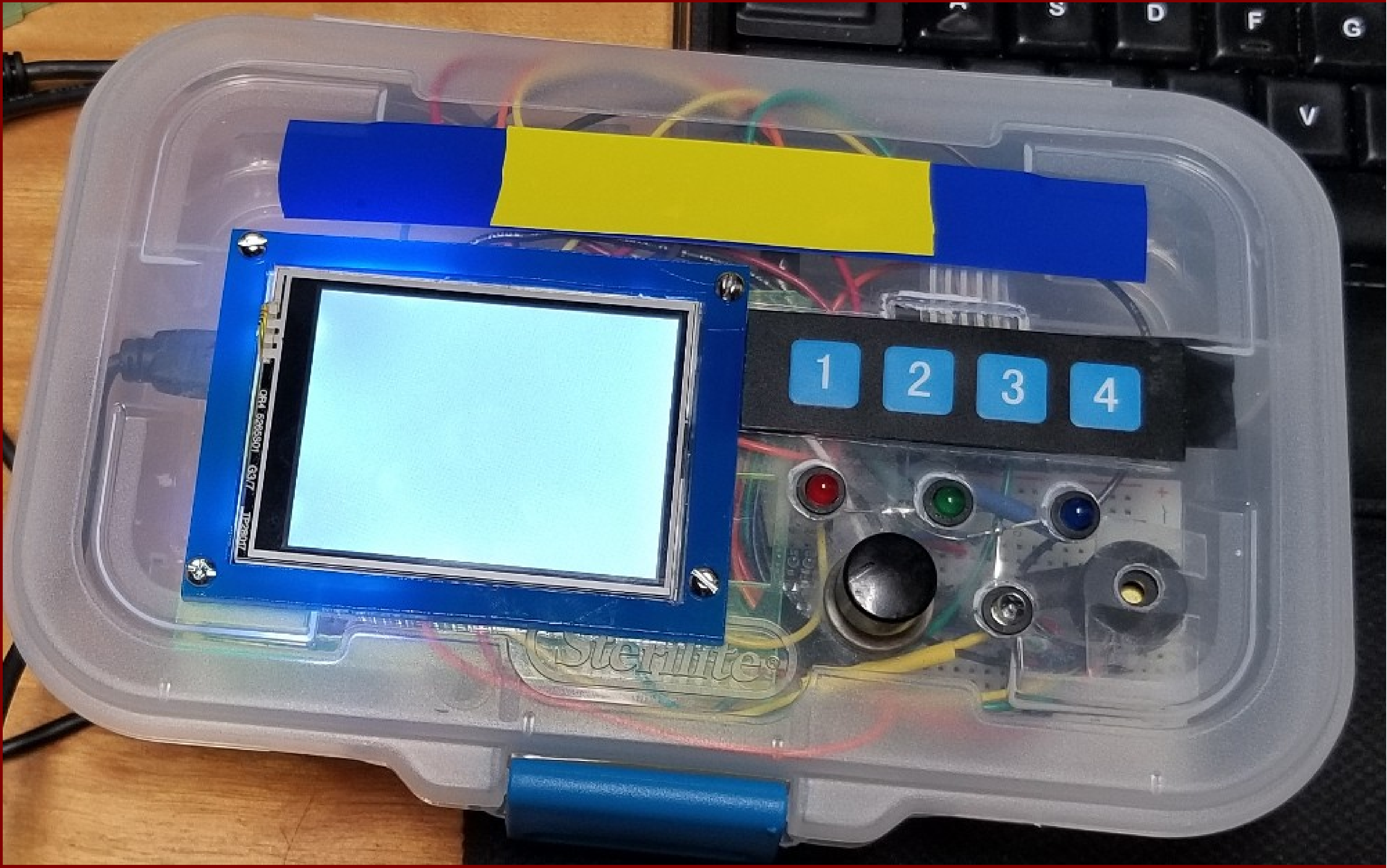
- After three back-to-back startups, took a long sabbatical to do just as I pleased
- Looped back to a focus on electronics after childhood studies under IBM dad while engaging with area interest groups and community service work
- Doing embedded business as Apex Proto Factory
- Design, fab, software, consulting
- Also some teaching

Context

- Teaching an ad hoc software development/engineering course
- 90% Lab, 10% Lecture
- Created repository of software and a series of “lab kits” of Arduino-based hardware to explore embedded systems

LabKit 3.0

- 240x320 touchscreen, Uno, buttons, LEDs, piezo, I2C bus brought out



Composing Asynchronous Programs

- How hard is it to make an Arduino rub its (figurative) stomach, pat its head, and sing a song in response to a cue, all at the same time and with different time measures?
- How hard is it to make it reliable?
- How hard is it to change?

The Right Tool for the Job

- Arduinos are great for simple tasks, but we sometimes need to push them hard
- In industry, standard practice for an application involving a lot of tummy rubbing and needing very reliable responses to cues (e.g. inside a car engine) is to use a “Real Time Operating System” (RTOS)
  - Preemptive thread (task) switching
  - Priorities
  - Real and virtual oodles of other stuff like resource management

But Arduinos Are Memory-poor

- Uno has 2048 bytes data, 32k code
  - Preemption requires state save/restore, typically with multiple stacks: out of the question
  - A compromise is needed
- Cooperative multi-tasking but with one asynchronous mechanism

What Is a Task in This Context?

- A task is a managed function call with finite work per invocation that always returns. It can be invoked at a set time, in response to an event signaled by another task or interrupt handler, or just whenever it next gets a turn in a round-robin fashion
- Wikipedia “cooperative multitasking”

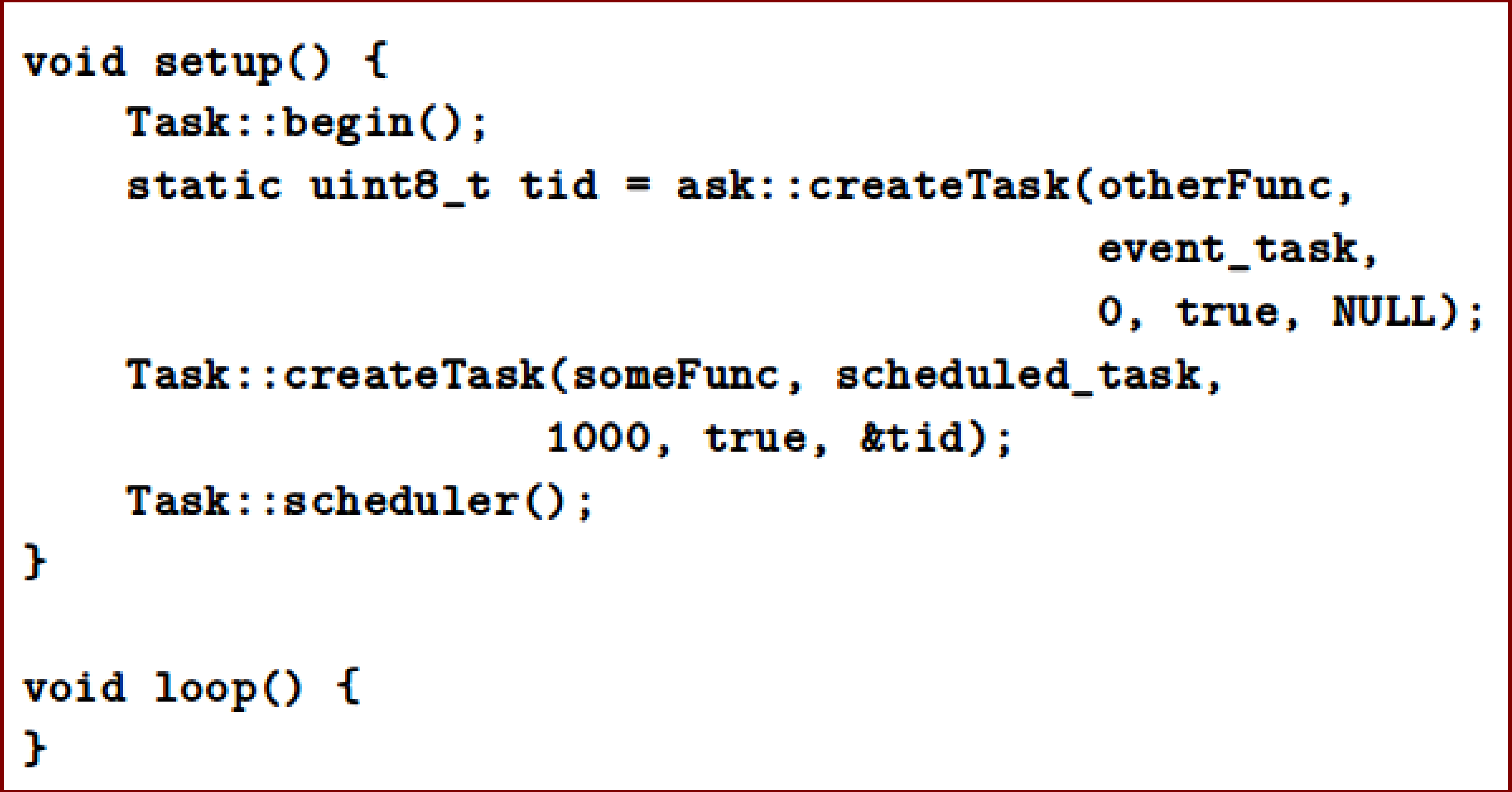
A Simple Task Library

- An appplication creates tasks, then starts a scheduler that never returns
- Three task flavors:
  - Regular: runs whenever it can
  - Scheduled: runs after a set time
  - Event: runs after an event
- Three states:
  - Executing
  - Runable: waiting for turn to execute
  - Pending: waiting for time or event

Creating a Task

- uint8_t createTask(void_func func, task_type type, uint32_t wait_milliseconds, bool keepalive, void *local);
- Returns ID
- Function to call, type as per previous slide
- Nonzero wait relevant for scheduled
- When keepalive is false, task is destroyed after next execution
- Multiple tasks sharing same function can have different task-local data

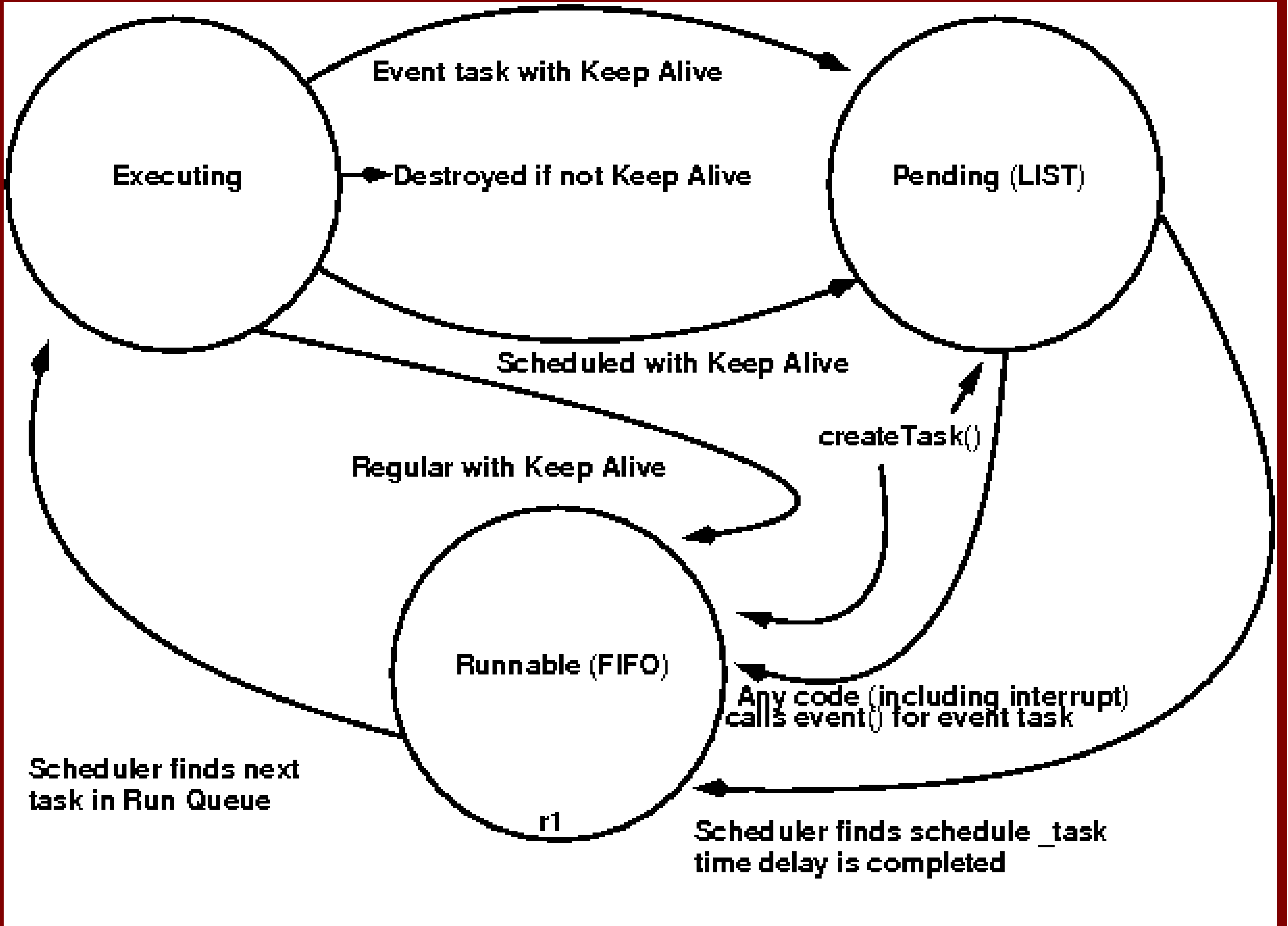
Typical Program Set Up



```
void setup() {
    Task::begin();
    static uint8_t tid = ask::createTask(otherFunc,
                                         event_task,
                                         0, true, NULL);
    Task::createTask(someFunc, scheduled_task,
                    1000, true, &tid);
    Task::scheduler();
}

void loop() {
}
```

How it Works



Executing

Pending (LIST)

Runnable (FIFO)

Event task with Keep Alive

Destroyed if not Keep Alive

Scheduled with Keep Alive

createTask()

Regular with Keep Alive

Any code (including interrupt)
calls event() for event task

Scheduler finds next
task in Run Queue

r1

Scheduler finds schedule _task
time delay is completed

Task State by Type

References and Q & A

- Public repository https://bitbucket.org/sugarpops/labkit
  - Subdirectories lib/Task and lib/NRingBuffer
- Email pete@soper.us